공학석사학위논문

정규화 학습을 이용한 심층 강화학습 기반 반도체 패키징 라인 스케줄링 기법의 강건성 향상

Enhancing Robustness of Deep Reinforcement Learning based Semiconductor Packaging Lines Scheduling with Regularized Training

2019 년 8 월

서울대학교 대학원

산업공학과

김 중 균

정규화 학습을 이용한 심층 강화학습 기반 반도체 패키징 라인 스케줄링 기법의 강건성 향상

Enhancing Robustness of Deep Reinforcement Learning based Semiconductor Packaging Lines Scheduling with Regularized Training

지도교수 박 종 헌

이 논문을 공학석사 학위논문으로 제출함

2019 년 6 월

서울대학교 대학원

산업공학과

김 중 균

김중균의 공학석사 학위논문을 인준함

2019 년 6 월

위 원 장 조 성 준 (인)

부위원장 박 종 헌 (인)

위 원 장 우 진 (인)

초록

최근 고성능 전자 제품에 대한 수요가 높아지면서 다중 칩 제품 생산을 중심으로 반도체 제조공정이 발전하고 있다. 다중 칩 제품은 패키징 라인에서 공정을 여러 번 반복하는 재유입이 발생하게 되며, 공정 설비의 셋업 교체가 빈번히 일으키게 된다. 이는 반도체 패키징 라인의 스케줄링을 어렵게 만드는 주요한 요소이다. 또한, 반도체 패키징 라인은 제조공정 내,외적으로 다양한 변동 사항에 의해 생산환경이 빈번히 변화하며, 제조 현장에서는 스케줄링을 위해 요구되는 계산 시간이 매우 중요하기 때문에 신속한 스케줄 도출이 요구된다. 반도체 패키징 라인의 스케줄링 연구가 활발해지면서 전역 최적화를 목표로 하는 강화학습 기반의 스케줄링 연구가 늘어나고 있다. 강화학습 기반의 반도체 패키징 라인 스케줄링 연구는 그 활용 측면에서 다양한 생산환경 변화에 강건히 대응하며, 짧은 시간 안에 좋은 스케줄을 얻을 수 있어야 한다.

본 연구에서는 심층 강화학습 기반의 스케줄링 모델의 강건성 확보를 목표로 한다. 새로운 생산환경이 테스트로 주어졌을 때, 재학습을 수행하지 않고 성능의 큰 저하없는 심층 강화학습 기반 반도체 패키징 라인 스케줄링을 위한 정규화 학습기법을 제안한다. 유연 잡샵 스케줄링 문제에 강화학습을 적용하기 위해 전체 공정 상황을 고려한 상태와 행동, 보상을 설계하였고, 심층 강화학습의 대표 알고리즘인 심층 Q 네트워크를 이용하여 스케줄링 문제를 학습하였다. 본 연구에서 제안하는 정규화 학습 기법은 4단계로 나누어 각 단계에서 여러 생산환경 변화가 반영된 문제의 일반성과 각 문제의 특수성을 학습하도록 설계하였다. 서로 다른 복잡도의 스케줄링 문제를 이용하여 실험을 진행하였으며, 룰 기반 및 심층 강화학습 기반의 다른 스케줄링 모델에 비해 대체적으로 성능의 우수함을 검증하였다.

본 연구는 강화학습 기반의 스케줄링 연구에서 모델의 강건성에 연구의 초점을 맞춘 첫 연구이며, 본 연구의 결과는 실제 공장에서 연구의 활용성을 한층 높여준 연구이다.

주요어: 반도체 패키징 라인, 유연 잡샵 스케줄링, 강건성, 심층 강화학습,
정규화 학습
학번: 2017-29546

목차

표 목차

그림 목차

제 1 장 서론

1.1 연구 배경 및 동기

최근 휴대용 전자 장비들의 대중화가 이뤄지면서 반도체 제조공정은 다중 칩 제품(multi-chip product) 생산을 중심으로 발전하고 있다 [1]. 다중 칩 제품은 쌓아올리는 칩의 수에 따라 기존 단칩 패키지 대비 많게는 수십 배의 복잡한 공정을 거쳐야 한다 [2]. 다중 칩 제품은 반도체 패키징 라인 중 DA(die attach)작업과 WB(wire bonding) 작업의 반복적인 재유입(re-entrance)이 발생하며, 이는 반도체 패키징 라인의 복잡도를 증가시킨다. 뿐만 아니라 다른 종류의 다중 칩 제품을 생산하기 위해서는 상당한 시간의 셋업 교체가 요구된다. 반도체 패키징 라인의 복잡도 증가와 셋업 교체 시간의 소요는 라인의 설비 가동률을 비롯한 여러 핵심 성과 지표(key performance indicator)를 저하시키는 요인으로 작용하였다 [3, 4].

반도체 패키징 라인은 시장 수요의 높은 변동성으로 인해 각 제품들의 생산요구량이 빈번히 변한다 [5]. 또한 설비의 고장, 유지 보수 등으로 인한 작업 별 설비의 비율, 스케줄 시작 시점에서의 설비 초기 셋업 상태 등이 빈번히 변화한다. 이러한 빈번한 생산환경 변화를 반영하지 않은 스케줄은 반도체 패키징 라인의 KPI를 저하시키는 요인으로 작용한다. 그러므로 생산환경 변화에 적절하게 대응하는 스케줄을 도출하는 것이 중요하다.

반도체 패키징 라인의 DA와 WB 작업은 유연 잡샵 스케줄링 문제(flexible job shop scheduling problem)로 모형화 할 수 있으며, 이를 해결하려는 다양한 연구가 진행되었다. 대표적으로 메타휴리스틱 기법을 이용한 스케줄링 연구가 진행되었다 [6–8]. 하지만

스케줄링을 위해 요구되는 계산 시간이 현장에서 매우 중요한 요소임을 감안하면 [9], 제안된 메타휴리스틱 기법의 좋은 해를 찾기 위해서는 오랜 계산시간이 소요되므로, 현장에서의 기법 적용 관점에서 적용하기엔 무리가 있으며, 짧은 계산시간이 소요되는 방법이 필요하다.

전역 최적화를 목표로 하는 강화학습의 특성을 이용하여 스케줄링 문제를 해결하고자는 연구들이 지속적으로 진행되어 왔다 [10–20]. 최근 심층 신경망을 활용한 심층 강화 학습 기법이 제안되면서 [21], 이를 스케줄링에 성공적으로 적용한 연구가 늘어나고 있다 [22]. 하지만 기존의 강화학습 기법을 활용한 스케줄링 연구는 생산환경 변화 대응에 연구의 초점을 두고 있지 않다. 반도체 패키징 라인에서 생산환경이 빈번히 변화하는 특성과 기법의 현장 적용에 있어 계산 소요 시간이 중요하다는 점에서 재학습 없이 변화에 강건하게 대응할 수 있는 강화학습 모델 설계가 필요하다. 여기서 언급한 모델의 강건성이란 학습되지 않은 테스트 상황에서도 '비슷한' 성능을 보이는 모델을 의미한다 [23].

기존의 강화학습은 모델의 강건성 측면에서 한계를 가지고 있다. 즉, 학습하지 못한 새로운 문제상황에서 기존에 학습된 강화학습 모델은 일반적으로 성능이 떨어진다 [24]. 이는 기계학습 중 지도학습에서의 과적합 현상과 유사하며, 이를 해소하기 위한 대표적인 방법이 정규화(regularization)를 활용하는 방법이다. 마찬가지로, 강화학습에서 모델의 강건성을 확보하기 위한 시도로 정규화를 활용한 연구가 있으며 [25], 주어진 문제 상황인 반도체 패키징 라인 스케줄링 문제에서 정규화 기법을 활용하여 강화학습 기반 스케줄링 모델의 강건성을 확보하고자 한다.

1.2 연구 목적

본 연구에서는 학습하지 않은 새로운 생산환경이 테스트로 주어졌을 때, 재학습을 수행하지 않고 성능의 큰 저하 없는 심층 강화학습 기반 반도체 패키징 라인 스케줄링을 위한 정규화 기법을 제안한다. 정규화가 포함된 심층 강화학습 모델이 다양한 생산환경에서 학습되어 여러 변화 상황에 대응할 수 있는 반도체 패키징 라인 스케줄링 모델로써 역할을 한다. 이를 통해 유동적인 시장 수요, 설비 고장 등의 상황에 따른 생산환경 변화에 신속히 대응할 수 있는 스케줄을 도출하는 것을 목적으로 한다.

1.3 연구 대상 정의

본 연구에서는 [6, 26]에서 설명되어 있는 반도체 패키징 라인의 DA와 WB작업의 스케줄링 문제를 다룬다. 아래에 정의하는 용어는 기존 연구인 [6, 26]의 용어 표기법을 참고하였다. 반도체 패키징 라인에서 하나하나의 제품은 랏(lot)단위로 그룹지어 설비에서 작업된다 [27]. 본 연구에서 고려하는 문제는 N_M개의 설비가 있으며 k번째의 설비는 M_k로 표현된다. 각 설비는 같은 시간에 한 개의 공정만 수행할 수 있다. 잡 타입(job type)은 앞서 언급한 제품의 종류를 의미하며, N_J만큼 존재한다. i번째 잡 타입은 J_i로 표현하며, J_i는 문제에 따라 정의된 생산요구량이 있다. J_i의 j번째 공정 종류(operation type)은 $O_{i,j}$로 표현하며, 모든 공정의 수를 N_O로 나타낸다. 각 공정은 정해진 순서에 따라 진행되며, 같은 종류의 잡 타입이면 공정의 순서는 동일하다. 또한, 각 공정이 설비에서 작업이 진행되면 중간에 종료되는 경우는 고려하지 않는다.

작업 공정 종류가 $O_{i,j}$인 공정은 $O_{i,j}$의 대체 설비(alternative machines)에 속하는 어떠한 설비도 작업이 가능하다. 이러한 대체 설비의 집합을 $A(O_{i,j})$라 하자. 본 연구에서는 공정이 DA와 WB로 나뉘기 때문에 DA작업에 해당하는 설비는 하나의 대체 설비 집합을 이루고, WB작업에 해당하는 설비들은 하나의 대체 설비 집합을 이룬다.

설비가 어떠한 공정을 수행하고자 할 때, 설비는 해당 공정의 종류로 셋업되어 있어야 한다. 예를 들어, 공정 종류가 $O_{i,j}$인 임의의 공정이 m_k에서 수행되기 위해서는 m_k의 셋업 상태(setup status)가 $O_{i,j}$이어야 한다. 만약 설비가 해당 공정 종류로 셋업되어 있지 않다면, 설비는 해당 공정으로 셋업 교체(setup change)를 수행하여야 한다. 이는 셋업 교체 시간이 소요되며, 그 동안은 다른 어떠한 공정을 진행할 수 없다. m_k에서 셋업을 $O_{i,j}$에서 $O_{i',j'}$으로 바꾸기 위해서는 $s_{i,j,i',j'}$의 시간이 소요된다. 일반적인 유연 잡샵 문제에서는 $i = i'$이고, $j = j'$이면 $s_{i,j,i',j'}$는 0이며, 아닐 경우에는 양수의 값을 가진다. 하지만 본 연구에서는 잡 타입만 같다면 공정의 차수에 상관없이 셋업 교체

시간이 소요되지 않는다. 즉, $i = i'$이면 $s_{i,j,i',j'}$는 0이며, 아닐 경우에는 양수의 값을 가지는 것으로 제한한다. 공정 순서에는 독립적이므로 이후부터는 셋업 교체 시간을 $s_{i,i'}$로 표현한다. 또한, DA작업을 수행하는 설비간 셋업 교체 시간이 동일하며, 마찬가지로 WB작업을 수행하는 설비간 셋업 교체 시간이 동일하다.

본 연구에서 가정하는 사항은 다음과 같다. 랏의 이동시간은 고려하지 않는다. 또한, 특정 잡의 특정 공정의 작업 시간은 동일하다. 또한 DA작업을 수행하는 설비 내에서의 셋업 교체 시간은 잡 타입에 관계없이 동일하며, WB작업을 수행하는 설비 또한 마찬가지다. 모든 설비는 시점이 0부터 가동되며, 모든 작업은 시점 0부터 수행된다. 원재료를 저장하는 카세트 스토커(cassette stocker)와 중간생산물을 저장할 수 있는 스토커(stocker)의 크기는 무한하다고 가정한다.

본 연구의 목적함수는 전체공기이며, 일반적으로 C_{max}로 표현하므로 본 연구에서도 이후의 목적함수는 C_{max}로 표시한다.

1.4 연구 내용 및 구성

반도체 패키징 라인 스케줄링을 위해 심층 강화 학습 기법 중 심층 Q 네트워크를 활용한다. 심층 Q 네트워크는 구글의 DeepMind에서 제안한 대표적인 심층 강화 학습 기법 중 하나이다 [21]. 심층 Q 네트워크를 활용하기 위해 상태, 행동, 보상을 정의하고 기 개발된 시뮬레이터를 기반으로 스케줄링 문제를 학습한다. 시뮬레이터는 앞서 언급한 연구 대상 정의 및 제약사항을 반영하여 기존의 [27]에서 제시한 이산 사건 기반 시뮬레이터(discrete event based simulator;DEBS)를 본 연구에 적합하도록 수정하여 사용하였다. 시뮬레이터의 역할은 반도체 패키징 라인을 모사하여 주어진 생산요구량을 생산하면서 발생 할 수 있는 데이터를 생성하고, 이를 강화학습의 학습 데이터로 제공하는 역할을 한다. 또한 강화학습 모델과 연결되어 환경의 역할을 한다.

본 연구에서 제안하는 정규화 학습 기법은 크게 4단계로 나뉘어진다. 우선 심층 Q 네트워크가 학습될 때, 네트워크의 모든 층이 학습되는 단계, 이어서 심층 Q 네트워크 중 마지막 층인 Q층만 학습하는 단계, 그리고 정규화 가중치를 학습하는 단계로 이뤄진다. 앞서 3단계의 학습이 완료되면, 최종적으로 학습된 정규화 가중치를 이용하여 새로운 Q층을 학습하는 단계로 학습이 완료된다. 제안된 기법으로 학습된 모델을 다양한 스케줄링 기법들과 성능비교 함으로써 제안 기법의 우수함을 검증하고자 한다.

본 논문은 다음과 같이 구성된다. 2장에서는 본 연구에서 적용하고자는 기법인 강화학습과 정규화에 대해서 배경이론을 설명하고, 기존의 연구결과를 논의한다. 3장에서는 강화학습 기반 반도체 패키징 라인 스케줄링 방법에 대해서 소개한다. 4장에서는 본 연구의 핵심인 강화학습 강건성 확보를 위한 정규화 학습 기법을 제안한다. 5장에서는 제안된 기법의 성능을 실험 결과를 통해 확인한다. 마지막으로 6장에서는 본 연구의 결론과 한계점 및 향후 연구 방향에 대해서 논의한다.

제 2 장 배경이론 및 관련연구

2.1 배경이론

2.1.1 심층 강화학습

강화학습은 기계 학습의 한 영역으로, 어떤 주어진 환경 안에서 정의된 에이전트가 현재의 상태를 기반으로 보상을 최대화하는 행동을 하도록 유도하는 방법이다. 에이전트가 현재의 상태에서 행동을 하면 행동의 결과를 다시 환경으로부터 피드백을 받게된다 [28]. 이러한 행동의 의사결정 속에서 에이전트는 보상을 최대화하는 행동의 연속을 선택한다. 이러한 연속적인 의사결정을 진행하는 강화학습의 특성상 스케줄링에 활용될 수 있다. 아래의 그림 2.1 [28]은 강화학습의 기본적인 상호작용 과정을 보여준다. 강화학습에서의 환경은 마코프 결정 과정(markov decision process; MDP)으로 주어진다. MDP에서는 (S, A, R, P) 로 구성된다. 여기서 S는 상태, A는 행동, R은 보상, P는 전이확률을 의미한다. 아래 그림2.1에서 알 수 있듯이, 에이전트는 주어진 환경에서 시점 t의 상태인 S_t를 인지한다. 이를 기반으로 행동인 A_t를 행한다. 환경은 에이전트가 한 행동을 바탕으로 다음의 상태인 S_{t+1}과 시간 t 시점의 보상인 R_t를 에이전트에게 피드백한다. 일련의 피드백 과정을 통해 에이전트는 모든 보상의 합을 최대화하는 방향으로 학습을 진행한다.

일반적으로 MDP의 경우 환경에서의 전이확률인 P를 알고 있다고 전제한다. 본 연구에서 사용하는 강화학습 알고리즘인 Q 러닝은 전이확률인 P를 모르는 모델 자유(model-free) 방법론이다 [29]. Q 러닝은 주어진 상태에서 주어진 행동을 수행하는 것이 가져다줄 효용의 기대값을 예측하는 함수인 Q 함수를 학습함으로써 최적의 정책을 학

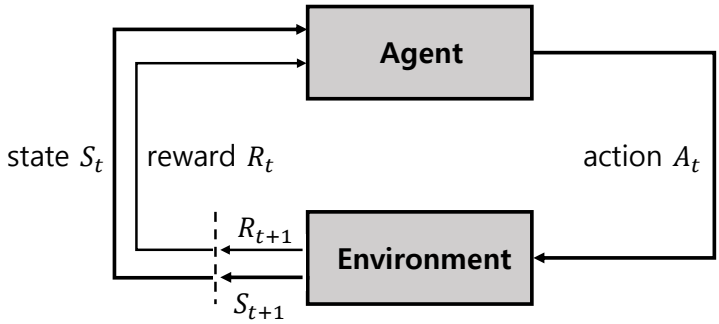

Figure 2.1: 강화학습 상호작용 기본구조

습한다. 여기서 말하는 정책은 어떤 행동을 행할지에 대한 규칙을 의미한다. Q 함수를 학습하고나면 주어진 상태에서 어떠한 행동을 했을 때 최대의 보상을 받을 수 있는지 최적의 정책을 유도할 수 있다. Q 러닝은 아래의 수식 2.1으로 표현된다. 여기서 Q는 행동 가치 함수로서 해당 시점에서 특정 행동을 했을 때 얻을 수 있는 가치를 의미하며, 이는 최적의 행동 가치 함수인 q_* 을 추정하는 함수이다. α는 학습률을 의미하며, γ 는 시간에 대한 할인율을 의미하며 이는 현재 얻는 보상이 미래에 얻는 보상보다 얼마나 더 중요한지를 나타내는 값이다. Q 함수의 학습은 현재 상태(S_t)에서 특정 행동(A_t)를 취했을 때 발생하는 즉각적인 보상(R_{t+1})과 다음 시점(S_{t+1}) 에서 예상되는 최대 Q 값의 합에서 현재의 Q 값을 뺀 차이만큼 업데이트를 진행한다 [28].

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)] \qquad (2.1)$$

본 연구에서 적용한 심층 강화학습 기법은 심층 Q 네트워크이다. 이는 구글의 Deepmind에서 제시한 기법으로, 기존의 Q 러닝에서 행동 가치 함수를 의미하는 Q 함수를 심층신경망을 적용하여 비선형 구조의 학습을 가능하게 만들었다 [21]. 심층 Q 네트워크는 Q 함수를 심층신경망으로 학습시키는 점 이외에 크게 두 가지의 기여를 통해 성공적인 연구가 가능하였다. 첫 번째로 경험 재현 버퍼(experience replay buffer)에서

데이터를 저장하여 미니 배치(mini batch)의 수만큼 데이터를 임의로 추출함으로써 인접한 학습 데이터 간의 상관성을 극복하였다. 두 번째는 심층신경망의 학습의 안정성을 확보하기 위해 온라인 네트워크(online network)와 타겟 네트워크(target network)를 따로 구분하고, 주기적으로 타겟 네트워크를 업데이트 함으로써 학습의 수렴이 어려운 심층신경망의 학습 안정성을 확보했다. 제안된 심층 Q 네트워크는 Atari 게임에서 게임의 이미지를 입력으로 하여 각 게임의 에이전트를 행동하도록 학습하였다. 다양한 게임의 실험을 통해 사람보다 더 높은 점수를 얻는 게임 에이전트를 학습하였다.

2.1.2 정규화

정규화란 기계학습, 특히 딥러닝에서 핵심적인 요소 중 하나이며 제한된 학습 데이터나 불완전한 최적화 과정 중에서 학습되지 않은 데이터에 대해 일반화 할 수 있도록 도와주는 기법이다 [30]. [31]에서 정의한 바에 따르면, 정규화는 테스트 데이터 셋에 대해 좋은 성능을 낼 수 있도록 모델의 일반화를 목표로 하는 기법이다. 최근 많은 모델에서 신경망을 함수로 사용한다. 신경망을 함수로 생각하고, $f_w : x \mapsto y$ 이며, 여기서 학습하는 매개변수는 $w \in W$ 이다. 신경망을 학습한다는 의미는 주어진 손실 함수인 $L : W \to \mathbb{R}$ 을 최소화하는 매개변수인 w^*를 찾는것이다. 아래는 이를 나타낸 수식이다. 수식 2.2는 최적의 매개변수를 찾는 것을 나타내며, 수식 2.3은 정규화가 포함된 손실 함수를 의미한다. 일반적으로 정규화 손실은 함수 f_w의 매개변수를 이용한다. 여기서 t는 종속변수의 값인 타겟을 말하고 P는 데이터의 분포, E는 에러함수를 의미한다.

$$w^* = \text{argmin}_w \, L(w) \tag{2.2}$$

$$L = \mathbb{E}_{(x,t)\sim P}[E(f_w(x), t) + R(w)] \tag{2.3}$$

주로 활용되는 정규화 손실은 $L1$ 정규화와 $L2$ 정규화를 이용한다. 각각은 아래와 같다.

$$R_{L1}(w) = \|w\|_1 = (\sum_i |w_i|) \tag{2.4}$$

$$R_{L2}(w) = \frac{1}{2}\|w\|_2 = \frac{1}{2}(\sum_i w_i^2)^{\frac{1}{2}} \tag{2.5}$$

각 정규화는 손실함수에 포함되어 페널티(penalty)처럼 작용하여 신경망에서 학습되는 파라미터인 w가 너무 커지는 것을 방지한다. 이를 통해 기계학습 중 지도학습에서 학습 데이터 셋에 과적합(over-fitting)되는 것을 방지하여 테스트 데이터 셋에 성능을 높이는 역할을 한다.

2.2 관련연구

2.2.1 반도체 패키징 라인 스케줄링 연구

앞서 언급한 반도체 패키징 라인 중 DA작업과 WB작업은 유연 잡샵 스케줄링 문제로 모형화 가능하다. 이는 NP-hard 문제이며, 재유입이 존재하는 만큼 스케줄링의 복잡도가 높은 문제이다. 유연 잡샵 스케줄링은 두 가지의 서로 다른 의사결정이 이뤄져야한다. 첫 번째는 설비에 작업을 할당하는 라우팅(routing)이며, 두 번째는 설비에 할당된 작업들의 순서를 결정하는 시퀀싱(sequencing)과정으로 이루어진다 [7]. 비교적 최근의 연구들은 메타 휴리스틱 기법을 적용하여 문제를 해결하고자 하였다 [6, 7, 32]. 하지만 메타 휴리스틱의 경우 좋은 해를 찾기 위해서 오랜 시간이 소요되며, 이는 스케줄링 모델의 실제 적용에 있어 어려움을 야기한다. 메타 휴리스틱 이외에도 유전 알고리즘을 사용하여 유연 잡샵 스케줄링 문제를 성공적으로 적용한 연구가 있다 [33, 34].

2.2.2 강화학습 기반 스케줄링 연구

기존 연구들은 다양한 연구 문제를 대상으로 강화학습을 스케줄링에 적용해왔다. 설비의 형태에 따라 또는 생산공정의 유형에 따라 다양하게 강화학습을 적용하여 스케줄링 문제를 풀었고, 성공적인 연구를 진행했다. 설비의 형태 관점에서 살펴보면, [10]는 강화학습을 이용하여 단일 설비에서 디스패칭 룰을 선택하는 문제를 연구하였다. 다양한 디스패칭 룰에서 가장 좋은 룰을 선택하는 행동을 강화학습을 통해 의사결정하였다. [11]에서는 단일 설비의 동적 잡샵 스케줄링를 해결하고자 하였다. 앞서 제시된 두 가지 연구 모두 단일 설비에 적용한 연구 사례이다. 이와 달리 병렬 설비에 대해 강화학습을 적용하여 스케줄링 문제를 다룬 연구는 [12, 13, 18]이다. 생산공정의 유형에 따라 플로우샵, 잡샵, 유연 잡샵으로 구분할 수 있다. 플로우샵 공정에 적용된 연구는 [14, 19, 20]이 있다. [14]에서는 플로우샵에서 전체공기을 최소화하는 문제를 강화학습

으로 풀었다. [19]는 플로우샵에서 다양한 기계와 여러 유형의 제품이 존재할 때, 공정 시스템의 최적 스케줄링을 도출하기 위해 다중 에이전트 강화 학습 방식을 채택하였다. [20]에서는 다양한 작업 시간(processing time)을 갖는 설비에서 전체공기을 최소화하기 위해 듀얼 Q 러닝(dual Q learning)방법론을 도입하였다. 잡샵 스케줄링 문제에도 많은 연구가 진행되어 왔다. [15]연구는 강화학습 에이전트가 실시간의 잡샵 시스템에서 최우선 룰을 선택하는 행동을 하도록 학습시켰다. 이를 통해 동적 스케줄링 시스템에서의 강화학습 에이전트의 활용 가능성을 확인하였다. [16]에서는 잡샵 스케줄링을 분산된(distributed) 강화학습 에이전트를 통해서 해결하려 하였다. 사용된 강화학습 기법은 기존 연구에서 많이 사용하던 Q 러닝이 아닌 정책 기울기(policy gradient)기반의 모델을 사용하였다. [17]에서는 잡샵 환경에서 잡 라우팅에서의 의사결정을 Q 러닝 모델을 통해 의사결정을 시도하였다. 본 연구에서 제시하는 제조 환경인 반도체 패키징 라인은 유연 잡샵 스케줄링으로 모형화가 가능하며, [35]에서 유일하게 강화학습 기반으로 유연 잡샵 스케줄링 문제를 다루었으며 계층적인 접근과, 학습, 최적화를 조합하여 좋은 결과를 얻었다. 기존의 연구를 살펴보면, 비교적 단순한 플로우샵, 잡샵의 경우는 강화학습 기반으로 다양한 연구가 진행되어 왔지만, 더욱 어려운 공정 복잡도를 지니는 유연 잡샵 스케줄링의 경우에는 아직까지 강화학습 기반의 접근이 부족하다.

2.2.3 강화학습 강건성 연구

강화학습의 강건성이란, 학습하지 않은 테스트 문제에 대해서 '비슷한' 성능을 보이는 모델을 말한다 [23]. 강화학습에서도 모델의 강건성과 일반화를 위한 연구가 지속적으로 진행되어 왔다. 대표적으로 전이학습(transfer learning)을 이용한 연구들이 진행되어 왔다 [36–38]. 전이학습이란 학습 데이터와 테스트 데이터가 독립항등분포(independent and identically distributed)라는 가정을 완화하면서 소스 도메인(source domain)에서 타겟 도메인(target domain)으로의 지식을 전달하는 방법이다 [39]. 즉,

기존의 소스 도메인인 학습 데이터에서 학습된 지식이 새로운 모델을 학습할 때 전이되어 학습을 빠르게하며 모델의 예측력을 높이고자는 방법이다. [36]에서는 쉬운 작업(task)부터 배우면서 그 정보를 이용하여 어려운 작업에 학습된 보상 함수를 적용하였다. [37]연구는 여러 개의 에이전트가 있는 환경임을 가정한다. 여러 개의 에이전트가 있을 때, 서로 영향을 주기 때문에 전이학습을 통해서 이전에 배운 작업에 대한 지식을 전달하거나, 더 경험이 이뤄진 에이전트로부터 조언을 받는 형태의 학습을 제안했다. [38]에서는 소스 도메인과 타겟 도메인이 비슷하지 않은 상황을 가정하고, 룰 전이라는 기법을 제안하였다. 소스 도메인으로부터 정책을 요약하는 규칙을 배운 후, 이 규칙을 통해 타겟 도메인에서 빠르게 규칙을 배우는 것을 목표로 하였다. 전이학습은 새로운 타겟 도메인에 빠르게 학습을 시키는 목적의 학습 방법론이다. 그렇기 때문에 재학습이 필수적으로 수행된다.

전이학습 이외에도 소스 도메인과 타겟 도메인 간의 도메인 적응(domain adaptation)과 도메인 일반화(domain generalization)가 강화학습의 강건성과 일반화 연구의 큰 축이다. 도메인 적응에 가장 대표적이며, 초석이 되는 연구로 [40]에서 제안한 Model-Agnostic Meta-Learning(MAML)이다. MAML은 특정 모델에 한정되지 않고 분류, 강화학습 등에 적용될 수 있는 방법론이다. 이는 다양한 작업들을 메타 러닝 방식으로 학습하면서 도메인 적응에 적합한 초기점(initial point)를 찾는 방법이다. MAML로 학습된 모델은 새로운 타겟 작업에 대해서 학습을 진행 할 때, 학습을 통해 찾은 초기점을 시작으로 적은 수의 기울기 업데이트(gradient update)를 통해서 빠르게 좋은 성능을 보이는 모델로 학습해나가는 것을 제안했다. [41]연구에선 여러 가지의 작업 학습(multi-task learning)과 도메인 적응을 이용하여 여러 가지 게임들에 강화학습을 적용하였다. [42]에서는 도메인 적응연구를 강화학습의 대표적인 벤치마크 데이터인 Atari 게임에 적용하였다. 타겟 작업에 대해 은닉 피처 표현(hidden feature represen-

tation)을 초기화하고, 상태 표현을 전이학습을 통해 도메인간의 전이를 학습하였다. 도메인 적응에 활용된 방법론들 또한 새로운 도메인 작업에 학습을 빠르게 하기 위한 방법들로, 재학습을 필요로 한다. 도메인 일반화에 관한 연구는 [43, 44]가 있다. [43]에서는 MAML연구와 비슷하게 특정 모델에 국한되지 않는 model-agnostic한 메타 러닝 도메인 일반화 방법론을 제시하였다. [44]는 의사 결정 트리를 이용한 강화학습으로 지도학습을 통해 상태에 따른 동작의 상대적 효과를 일반화하여 모델을 학습하는 연구를 진행하였다.

제 3 장 강화학습 기반 반도체 패키징 라인 스케줄링

본 절에서는 강화학습 기반의 반도체 패키징 라인 스케줄링의 설계와 강화학습 스케줄링 에이전트가 학습 및 테스트되는 과정에 대해서 소개한다. 기본적인 강화학습 과정은 [45]에서 제시하는 학습 과정과 유사하다. 하지만 알고리즘과 과정의 세부사항이 다르기 때문에 본 절에서 자세하게 강화학습 학습과정을 다룬다. 이를 다루기 위해서는 3절에서 언급한 반도체 패키징 라인을 모사하는 시뮬레이터와 강화학습 에이전트가 어떻게 연결되는지, 스케줄링 의사결정은 어떻게 이루어지는지, 스케줄링 에이전트를 학습하기 위해 강화학습의 설계는 어떻게 이루어지는지를 순서대로 소개한다. 본 연구에서의 의사결정이란 설비의 셋업 교체에 관한 의사결정을 의미한다.

아래의 그림 3.1은 본 연구에서 적용하는 강화학습 기반의 스케줄링의 구성도를 나타낸다. 크게 학습 단계와 테스트 단계로 나뉜다. 학습 단계는 주어진 학습 스케줄링 문제를 입력받으면 모든 과정은 시뮬레이터 안에서 이루어진다. 학습 스케줄링 문제는 기 정의된 잡 타입 별 생산요구량, 설비의 구성, 초기 셋업 등으로 구성된다. 시뮬레이터 내에는 이산 사건들을 처리하고, 강화학습 에이전트로부터 설비의 셋업 의사 결정을 전달 받아 처리하는 의사결정 처리모듈이 있으며, 강화학습 에이전트를 학습하는 스케줄링 에이전트 학습모듈이 존재한다. 또한, 이 둘과 별개로 의사결정 처리모듈로부터 학습 데이터를 저장하여 이를 학습모듈에 전달해주는 경험 재현 버퍼로 구성되어있다. 의사결정 처리모듈은 반도체 패키징 라인을 모사하기 위해 정의된 여러가지 사건들을 시간 순서대로 처리하며, 설비에서 셋업 의사 결정이 필요할 때 온라인 Q 네트워크에게 상태를 전달하고 결정된 의사결정인 행동을 받아 이를 이행한다. 연속된 의사결정을 통해 만들어진 (상태, 행동, 보상, 다음 상태)을 트랜지션(transition)라고 하며, 이를 학습

데이터로 경험 재현 버퍼에 저장한다. 스케줄링 에이전트 학습모듈은 [21]에서 제시한 심층 Q 네트워크와 그 학습 방법을 차용하여 구성하였다. 스케줄링 에이전트 학습모듈은 매 의사결정이 필요할때 마다 호출되어 학습된다. 호출될 때 마다 경험 재현 버퍼로부터 일정 크기의 미니배치만큼의 임의로 추출한 학습 데이터를 이용하여 온라인 Q 네트워크를 정의된 학습 알고리즘에 따라 학습한다. 주기적으로 타겟 Q 네트워크의 가중치는 온라인 Q 네트워크의 가중치로 대체되며, 이를 통해 심층 Q 네트워크의 수렴 안정성을 도모하였다 [21]. 학습 단계에서 학습이 완료되면, 테스트 단계에서는 학습된 온라인 Q 네트워크를 이용하여 테스트 스케줄링 문제에서 의사결정을 진행한다.

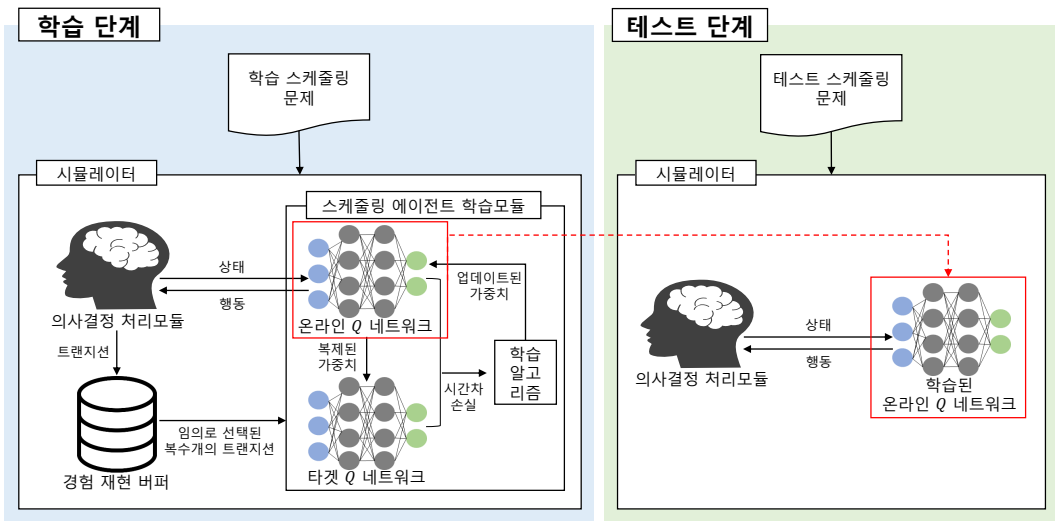

Figure 3.1: 강화학습 기반 스케줄링 구성도

3.1 강화학습 기반 스케줄링 의사 결정

강화학습에서의 환경 역할은 시뮬레이터에서 담당한다. 의사결정은 심층 Q 네트워크 중 온라인 Q 네트워크를 통해 이루어진다. 작업을 진행할 랏과 설비를 선택할 때, 환경인 시뮬레이터는 에이전트인 온라인 Q 네트워크에게 상태를 전달한다. 상태를 전달받은 에이전트는 결과물인 행동을 다시 시뮬레이터에 전달하고, 그 행동을 기반으로 의사결정 하여 다음 의사결정 시점까지 사건들을 처리한다. 의사결정이 필요한 시점은 모든 설비들 중 비어있는 설비가 존재하면서 대기 중인 작업이 가능한 랏이 있는 시점이다. 하나 또는 여러 개의 설비가 작업이 끝나고, 해당 시점에 대기 중인 랏이 있을 때, 에이전트에게 의사결정을 요청한다.

본 연구에서는 의사결정을 내리는 에이전트는 1개만을 이용한다. 즉, 다루고 있는 DA공정과 WB공정별로 따로 의사결정 에이전트를 구분하지 않고 하나의 에이전트를 이용하여 모든 의사결정을 진행한다. 하나의 에이전트를 사용하므로, 환경으로부터 생성되는 상태 또한 지역적인 상태가 아닌 공장 전체의 전역적 상태를 입력받는다. 그러므로 상태는 모든 공장을 잘 표현할 수 있는 방식으로 채택한다. 아래의 그림 3.2는 DA와 WB작업 공정에서의 랏의 흐름을 도식화하고, 환경과 에이전트간의 상호작용을 나타낸다. 랏의 흐름은 이전 연구인 [27]에서 제시한 랏의 흐름을 참고하였으며, 앞서 언급한 대로 상태는 특정 공정 또는 특정 설비에 관한 정보가 아니라 모든 공정과 모든 설비, 모든 잡 타입들의 정보를 표현하여 전역 상태를 구성하고, 이를 통해 하나의 에이전트의 의사결정으로 스케줄링 문제를 해결한다.

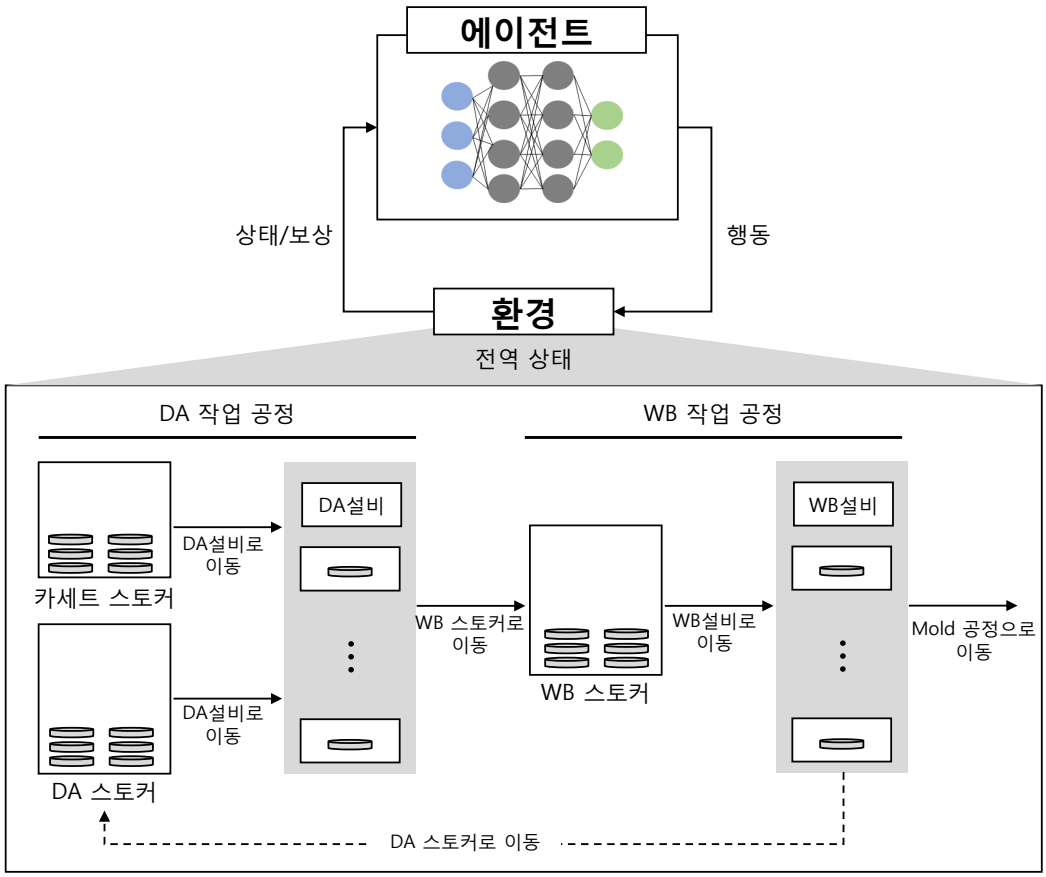

Figure 3.2: 공정 별 랏 흐름과 환경과 에이전트 상호작용

3.2 상태, 행동, 보상 정의

강화학습에서는 MDP의 구성요소인 상태, 행동, 보상을 정의하는 것이 성능에 중요한 역할을 한다. t번째의 상태, 행동, 보상을 각각 s_t, a_t, r_t라 하자. 상태는 일련의 수치 벡터로 구성되고, 공장의 모든 정보를 활용하도록 설계되었다. 크게 작업과 관련된 정보, 설비와 관련된 정보, 작업 공정과 관련된 정보로 나뉜다. 작업과 관련된 정보는 그림 3.2에서 도식화 되어있는 공정별 랏의 흐름을 수치적으로 표현하였다. DA작업과 WB 작업 수행을 위해 대기중인 랏의 수, 현재 각 공정에서 작업중인 랏의 수, 모든 공정을 끝마친 랏의 수를 잡 타입 별, 공정 별로 따로 수치화하였다. 즉, DA작업과 WB작업 수행을 위해 대기중인 랏의 수, 현재 각 공정에서 작업중인 랏의 수를 나타내는 상태 벡터는 각각 N_O차원을 가지며, 공정을 끝마친 랏의 수를 나타내는 상태벡터의 차원은 N_J이다. 각 상태벡터는 모두 스케줄링 문제에 투입되는 랏의 총 개수로 나누어 모든 벡터의 숫자를 0과 1 사이로 표준화시켜주었다. 표준화를 하는 이유는 투입되는 랏의 개수가 변화하는 문제에서는 심층 신경망의 입력값인 상태벡터의 각 요소(element)들의 크기(scale)가 달라지므로 학습에 악영향을 미칠 수 있기 때문이다. 설비와 관련된 정보는 각 공정 별, 잡 타입 별 설비의 할당 비율과 의사결정 시점에서 비어있는 유휴 설비들의 공정 별, 셋업 타입 별 비율을 나타낸다. 각각의 상태벡터는 $N_J * 2$의 차원을 가진다. N_J에서 두 배가 된 이유는 공정이 DA,WB 두 공정이기 때문이다. 마찬가지로, 앞서 설명한 설비 관련 상태벡터는 비율이므로, 각 공정 별 설비 총 개수로 나눠주었다. 추가로 설비에 관한 정보로, 설비가 가질 수 있는 3가지의 상태인 유휴, 작업중, 셋업 교체 중에 해당하는 설비들의 비율을 상태벡터로 채택하였다. 이는 총 3차원을 가진다. 마지막으로, 공정 관련 정보는 현재 의사결정을 내리는 공정이 DA공정인지 WB공정인지에 대한 정보로 원-핫 인코딩(one-hot encoding)으로 나타내었다. 이를 정리하면 아래의 표 3.1와 같다.

Table 3.1: 상태벡터 구성요소

상태 정보	설명	차원
작업 관련 상태	DA 또는 WB 설비에서 작업 수행을 위해 대기중인 랏의 잡 타입 별 공정 별 비율	N_O
	DA 또는 WB 설비에서 작업중인 랏의 잡 타입 별 공정 별 비율	N_O
	모든 공정을 끝마친 랏의 잡 타입 별 비율	N_J
설비 관련 상태	각 공정 별,잡 타입 별 설비 할당 비율	$N_J * 2$
	의사결정 시점에서 비어있는 유휴 설비들의 공정 별, 셋업 타입 별 비율	$N_J * 2$
	유휴, 작업중, 셋업 교체 중에 해당하는 설비들의 비율	3
공정 관련 상태	DA 또는 WB 공정에 대한 표현(one-hot encoding)	2

행동은 강화학습 에이전트가 상태를 기반으로 선택해야 하는 의사결정이며, 본 연구에서는 설비의 셋업 상태와 잡 타입의 조합을 행동으로 정의한다. 설비가 가질 수 있는 셋업 상태는 잡 타입의 개수인 N_J이다. 마찬가지로, 잡 타입 또한 N_J만큼의 종류를 가질 수 있으므로, 이들의 조합인 $(N_J)^2 * 2$ 만큼을 행동의 총 가지수로 정의할 수 있다. 두 배가 되는 이유는, 하나의 전역 에이전트가 DA와 WB공정 모두의 의사결정을 행하기 때문이다. 시뮬레이터에서 정의한 행동의 의사결정을 할 때, 제약사항이 존재한다. 의사결정을 대기 중인 랏의 잡 타입과 비어있는 설비의 셋업 타입이 항상 모든 잡 타입을 포함하진 않는 경우가 발생한다. 이 경우엔, 수행 가능한 행동만 진행해야 하며, s_t에서 수행 가능한 행동의 집합을 $F(s_t)$라 하고, a_t는 $F(s_t)$의 원소 중 하나이다. $F(s_t)$는 다음과 같이 구성할 수 있다. 만약 현재 DA공정의 의사결정이면, WB공정 의사결정에 해당하는 행동들은 $F(s_t)$에 포함되지 않는다. 또한, 현재 J_i인 잡의 j 공정에 해당하는 랏이 대기중에 있고, 비어있는 설비가 $O_{i,j}$를 작업할 수 있고, 셋업 상태가 J'_i이라면 $J_i \times J'_i$의 조합이 $F(s_t)$의 원소가 될 수 있다.

마지막으로 보상은 의사결정이 일어날 때, 해당 의사결정을 통해 발생할 수 있는 셋업 교체 시간을 음수로 보상을 얻는다. 즉, 에이전트가 수행 가능한 행동 집합인 $F(s_t)$의 원소 중 하나를 선택하고, 선택된 설비의 셋업 상태가 J'_i이고 선택된 랏의 잡 타입이 J_i일 때, $i = i'$이면 셋업 교체가 일어나지 않으므로 보상으로 0을 얻으며, 그렇지 않으면 $-s_{i,i'}$만큼을 보상으로 얻는다. 본 연구의 목적함수는 C_{max}의 최소화이다. 모든 잡의

작업 시간은 스케줄링 문제가 주어지면 정해지는 값이므로, C_{max}까지의 모든 설비의 손실시간, 즉, 모든 설비의 유휴시간과 셋업교체시간의 합을 최소화하면 앞서 정의한 목적함수를 달성할 수 있다. 그러므로, 본 연구에서는 셋업교체시간을 음수 보상으로 하여 스케줄링 에이전트가 목적함수를 달성 할 수 있도록 유도하였다.

3.3 강화학습 에이전트 학습과 테스트

3.3.1 심층 Q 네트워크 구조

아래 그림 3.3는 본 연구에서 사용한 심층 Q 네트워크의 구조를 나타낸다. 입력층의 차원은 학습 스케줄링 문제에 따라 다르며, 출력층의 차원 또한 문제에 따라 다르다. 은닉층의 경우 3개의 층으로 구성되어있고, 각 층은 64개의 노드(nodes)로 구성되어 있다. 마지막 은닉층을 제외하고 각 층의 활성 함수(activation function)은 ReLU(Rectifed Linear Unit)을 적용하였다. ReLU, $f(z) = max(0, z)$, 는 비선형 변환을 하도록 도와주며, 기존의 심층 신경망에서 기울기값이 사라지는 문제(vanishing gradient)를 완화시켜주었다 [46]. 마지막 은닉층과 출력층의 연결 부분은 완전 연결층(fully connected layer;FC)으로 구성하였다.심층 Q 네트워크는 온라인 Q 네트워크와 타겟 Q 네트워크 두 개를 갖는다. 이 두 개의 네트워크는 구조가 동일하다.

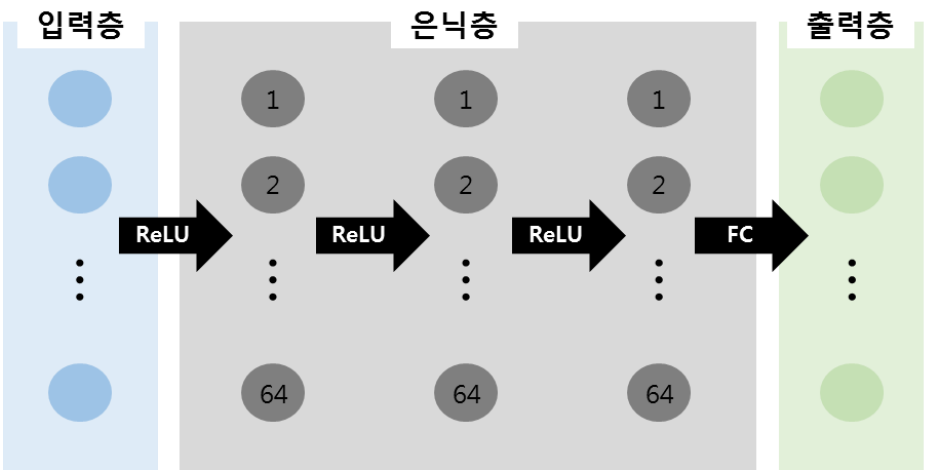

Figure 3.3: 심층 Q 네트워크 구조

3.3.2 강화학습 에이전트 학습 단계

스케줄링을 위한 강화학습 에이전트는 앞서 설명한 심층 Q 네트워크의 구조를 이용하여, [21]에서의 학습 알고리즘 수식과 표기법을 참고하여 학습을 진행한다. 기본적으로 심층 Q 네트워크의 입력은 상태벡터인 s_i이며, 출력값은 $Q(s_i, a_i)$로 나타내며, 이는 상태 s_i에서 행동 a_i가 행해질 때 기대 행동 가치를 의미한다. 아래의 그림 3.4의 알고리즘은 스케줄링을 위한 강화학습 에이전트 학습 절차를 나타낸다. 우선, 학습 스케줄링 문제와, N_S, N_{RB}, C를 입력한다. N_S는 주어진 학습 스케줄링 문제를 반복해서 풀면서 강화학습 에이전트 학습을 진행하는데, 총 시뮬레이션을 통해 스케줄링 문제를 푸는 횟수를 의미한다. N_{RB}는 경험 재현 버퍼의 크기를 나타낸다. C는 [21]에 제시된 파라미터로, 타겟 Q 네트워크의 가중치를 온라인 Q 네트워크의 가중치로 업데이트하는 주기를 나타낸다. 최종적으로 학습과정을 통해 도출되는 것은 심층 Q 네트워크, 구체적으로 온라인 Q 네트워크이다. 학습이 시작되면, 경험 재현 버퍼와 온라인, 타겟 Q 네트워크의 가중치를 초기화한다. 학습 과정은 알고리즘의 줄 3-28이 N_S 만큼 반복된다. 시뮬레이션이 시작하면 가장 먼저 주어진 스케줄링 문제대로 모든 잡과 설비가 초기화되며, 스케줄링 시간인 t와 의사결정 순서인 i가 0으로 초기화된다. 스케줄링 문제를 풀기 위해서, 시간은 마지막 공정이 끝나는 시각인 C_{max}까지 반복된다. 스케줄링 의사결정을 위해서는 t시점에 유휴 상태(idle)인 설비가 있어야하고 동시에 작업 대기중 랏이 존재해야 한다. 의사결정이 일어나는 공정에 해당하는 유휴 설비의 집합을 IM_t라 하고, 해당 공정의 작업 대기중인 랏의 집합을 WO_t로 나타내며, 이 두 집합을 먼저 구한다. 두 개의 집합에 원소가 모두 존재할 때, 심층 Q 네트워크의 행동인 설비의 셋업 상태와 잡 타입의 조합을 행할 수 있다. 앞서 구한 IM_t와 WO_t이 동시에 존재한다면 두 집합 중 하나라도 빈 집합이 될 때까지 의사결정을 반복한다. 강화학습 에이전트는 상태벡터인 s_i를 관측하고, 이를 바탕으로 행동인 a_i를 수행한다. 학습 단

계에서는 입실론 그리디(ϵ-greedy) 정책을 이용한다 [28]. a_i를 선택하는 방법은 아래의 수식 3.1과 같다. 여기서 random($\cdot$)함수는 s_i에서 수행 가능한 행동의 집합 중 임의로 선택하는 것을 의미한다. ϵ은 임의 행동이 선택될 확률을 의미한다.

$$a_i = \begin{cases} \text{random}\left(F(s_i)\right) & \text{with probability } \varepsilon \\ \underset{a \in F(s_i)}{\text{argmax}}\, Q(s_i, a; \theta) & \text{otherwise} \end{cases} \tag{3.1}$$

행동이 수행될 때 마다, 트랜지션을 경험 재현 버퍼에 저장한다. 경험 재현 버퍼는 크기가 N_{RB}로 고정되어 있으며, 데이터가 많아져 N_{RB}보다 커지게되면 가장 오래된 트랜지션을 제거하고 새로운 데이터가 추가된다. 경험 재현 버퍼에 트랜지션이 추가되면, 강화학습 에이전트는 기 정의된 미니배치 크기만큼 경험 재현 버퍼에서 임의로 트랜지션인 (s_j, a_j, r_j, s_{j+1})들을 추출하여 이를 통해 학습을 진행한다. 여기서 미니배치 크기를 N_{mini}라하자. 학습은 타겟 값과 Q 값의 차이를 이용하여 진행된다. 타겟 값은 수식 3.2로 구할 수 있다 [21]. 여기서 $\hat{\text{Q}}$는 타겟 Q 네트워크를 의미하고, θ^-는 타겟 Q 네트워크의 파라미터인 가중치를 의미한다. 마찬가지로 a'는 $F(s_{j+1})$의 원소여야 한다. 여기서 γ는 할인율을 의미한다 [28].

$$y_j = \begin{cases} r_j & \text{if } s_{j+1} \text{ is terminal} \\ r_j + \gamma \max\limits_{a'} \hat{Q}(s_{j+1}, a'; \theta^-) & \text{otherwise} \end{cases} \tag{3.2}$$

최종적인 강화학습 에이전트의 손실함수는 아래의 수식 3.3이다. 본 연구에서는 [47]에서 제시한 huber 손실함수를 사용한다. 손실함수를 경사 하강법 기반의 최적화 기법을 통해 에이전트를 학습해나간다.

$$L(\theta) = \frac{1}{N_{mini}} \sum_{j=1}^{N_{mini}} f\left(y_j, Q(s_j, a_j; \theta)\right) \tag{3.3}$$

$$f\left(y_j, Q(s_j, a_j; \theta)\right) = \begin{cases} \frac{1}{2}\left(y_j - Q(s_j, a_j; \theta)\right)^2 & \text{if } |y_j - Q(s_j, a_j; \theta)| < 1 \\ |y_j - Q(s_j, a_j; \theta)| - \frac{1}{2} & \text{otherwise} \end{cases} \tag{3.4}$$

위의 과정을 통해서 의사결정이 일어나면, 선택된 a_i에 따라 특정 잡 타입으로 셋업된 설비 중 가장 낮은 인덱스를 가진 설비(M_k)가 선택되고, 특정 잡 타입의 랏 중에서 작업이 제일 적게 남은 고차 작업이 진행될 랏($l_{O_{n,m}}$)이 선택된다. 선택된 설비와 랏 쌍은 기존의 유휴 설비 집합인 IM_t와 작업 대기 중인 랏 집합인 WO_t에서 제외된다. 위의 과정을 두 집합 중 하나라도 빌 때까지 반복한다. 이 과정이 끝나면, 시간 t를 작업이 끝나는 가장 빠른 시간으로 변경한다. 작업이 끝나면 유휴 상태로 돌아가는 설비가 다시 발생하므로, 의사결정을 이어갈 수 있다. 일련의 과정을 모든 작업이 끝날 때까지인 C_{max}까지 반복한다. 그리고 C번의 스케줄링 시뮬레이션마다 타겟 Q 네트워크의 가중치를 온라인 Q 네트워크의 가중치로 대체한다.

3.3.3 강화학습 에이전트 테스트 단계

테스트 단계에서는 학습 단계에서 학습된 심층 Q 네트워크를 이용하여 테스트 스케줄링 문제를 푼다. 본 연구에서는 학습되지 않은 문제의 성능을 개선하고자는 연구이므로, 테스트 스케줄링 문제는 학습 스케줄링 문제 대비 잡 타입 별 생산 비율, 공정별 설비 비율, 초기 설비 셋업 상태 등이 변화된 문제를 의미한다. 학습 단계와의 차이는 모든 행동을 입실론 그리디 방식을 사용하지 않고, 학습된 온라인 Q 네트워크의 출력값에서 가장 큰 기대 행동 가치를 도출하는 행동을 선택한다. 즉, 행동을 임의로 선택하지

않으므로 여러번 스케줄링 문제를 풀기위해 시도해도 동일한 결과가 도출된다. 선택된 행동은 설비의 셋업 상태와 잡 타입의 조합이므로, 이를 통해 특정 설비와 특정 랏을 선택한다. 선택된 설비와 랏은 의사결정 시간과 함께 스케줄 의사결정 리스트에 추가된다. 또한 학습에 필요한 트랜지션을 경험 재현 버퍼에 저장하는 부분과 손실함수를 구하여 기울기 업데이트를 하는 부분은 제외된다. 아래의 그림 3.5 알고리즘 은 테스트 단계를 통해 스케줄이 생성되는 것을 보여준다.

Algorithm 1 Training phase : training deep Q network agent for scheduling

Input: Scheduling problem,
N_S: Iterations of solving scheduling problem,
N_{RB}: Capacity of replay buffer,
C: Target network update frequency
Output: Q-network

Initialize : Initialize replay buffer to capacity N_{RB},
Initialize online Q network with random weights θ,
Initialize target Q network with random weights $\theta^- = \theta$
for *iteration* $\in (1,..,N_S)$ **do**
  Initialize jobs and N_M machines
  Initialize $t = 0$, $i = 0$
  while $t \leq C_{max}$ **do**
    Build IM_t and WO_t
    if $IM_t \neq \emptyset$ and $WO_t \neq \emptyset$ **then**
      repeat
        if $i = 0$ **then**
          Observe s_i
          Execute action a_i according to (3.1)
        else
          Observe s_i and r_{i-1}
          Execute action a_i according to (3.1)
          Store transition $(s_{i-1}, a_{i-1}, r_{i-1}, s_i)$ in replay buffer
          Sample random mini-batch of transitions from replay buffer
          Calculate loss L from (3.2)-(3.4)
          Perform a gradient descent step on L w.r.t θ
        end if
        Select a machine(M_k) which has lowest index in IM_t and a lot($l_{O_{n,m}}$) which m is highest in WO_t according to a_i
        $IM_t \leftarrow IM_t \setminus \{M_k\}$
        $WO_t \leftarrow WO_t \setminus \{l_{O_{n,m}}\}$
        $i \leftarrow i + 1$
      until ($IM_t = \emptyset$ or $WO_t = \emptyset$)
    end if
    set t to the earliest time when there is a finished operation
  end while
  Every C iteration, reset $\theta^- = \theta$
end for
return Q-network

Figure 3.4: 학습단계: 심층 Q 네트워크 학습

Algorithm 2 Test phase : scheduling with deep Q network agent

Input: Scheduling problem,
trained Q network
Output: Scheduling decision list

Initialize jobs and N_M machines
Initialize $t = 0$, $i = 0$
while $t \leq C_{max}$ **do**
  Build IM_t and WO_t
  if $IM_t \neq \emptyset$ and $WO_t \neq \emptyset$ **then**
    repeat
      Observe s_i
      Execute action $a_i = \underset{a \in F(s_i)}{\operatorname{argmax}} Q(s_i, a)$
      Append a scheduling decision composed of a machine(M_k) which has lowest index in IM_t and a lot($l_{O_{n,m}}$) which m is highest in WO_t according to a_i
      $IM_t \leftarrow IM_t \setminus \{M_k\}$
      $WO_t \leftarrow WO_t \setminus \{l_{O_{n,m}}\}$
      $i \leftarrow i + 1$
    until ($IM_t = \emptyset$ or $WO_t = \emptyset$)
  end if
  set t to the earliest time when there is a finished operation
end while
return Scheduling decision list

Figure 3.5: 테스트단계: 심층 Q 네트워크 테스트

제 4 장 강화학습 강건성 확보를 위한 정규화 학습 기법

4.1 정규화 학습 개요

본 절에서는 본 연구의 제안 기법인 강화학습 강건성 확보를 위한 정규화 학습 기법을 설명한다. 본 연구에서 제시하는 방법론은 딥러닝 모델의 일반성을 연구한 [48]에서 영감을 얻어 발전시켰다. 참고한 연구에서도 정규화를 이용하여 모델의 도메인 일반화를 시도하였다. 학습 과정에서 정규화의 가중치를 학습하여 다른 도메인에 대한 성능을 높이고자 하였으며, 이미지 인식과 감성 분류 분야에 적용하였다. 이전 연구에서는 강화학습에 적용하지 않았지만, 강화학습에서도 유사한 방법론을 적용할 수 있으며, [48]에서 제시한 학습 방식을 차용하지만 본 연구의 특성에 맞게 수정, 보완하여 제안한다.

정규화 학습하는 과정은 4절에서 설명한 강화학습 기반 스케줄링 에이전트 학습 과정과 맞물려있다. 아래 그림 4.1는 강화학습 기반 스케줄링 에이전트 학습 및 정규화 학습 과정과 이를 이용한 테스트 과정을 나타낸다. 4절에서 언급한 그림 3.1와의 차이는 학습 스케줄링 문제가 여러 개이며, 정규화가 학습 되는 과정이 시뮬레이터와 별개로 추가로 진행된다. 기존의 강화학습 에이전트는 시뮬레이터 내부에서 의사결정 처리모듈과 결합되어 학습 및 의사결정이 이루어졌다. 하지만, 정규화 학습 부분은 의사결정 시점과는 무관하며, 스케줄링 문제를 한번 다 풀게된 이후에 학습을 진행한다. 학습 데이터는 저장되어있는 경험 재현 버퍼에서 미니배치 만큼의 데이터를 임의로 추출하여 사용한다. 네트워크 업데이트에 이용되는 손실함수는 상황에 따라 기존의 손실함수인 시간차 손실함수를 적용하거나, 시간차 손실에 정규화 손실이 추가된 함수를 적용한다. 시뮬레이터에서의 학습과 시뮬레이터 외부에서의 정규화 학습이 반복적으로

진행되며, 정해진 일정 학습 횟수가 끝나면 최종적으로 온라인 Q 네트워크의 특정 층을 초기화하여 새롭게 추가 학습하여 모든 학습과정을 종료한다.

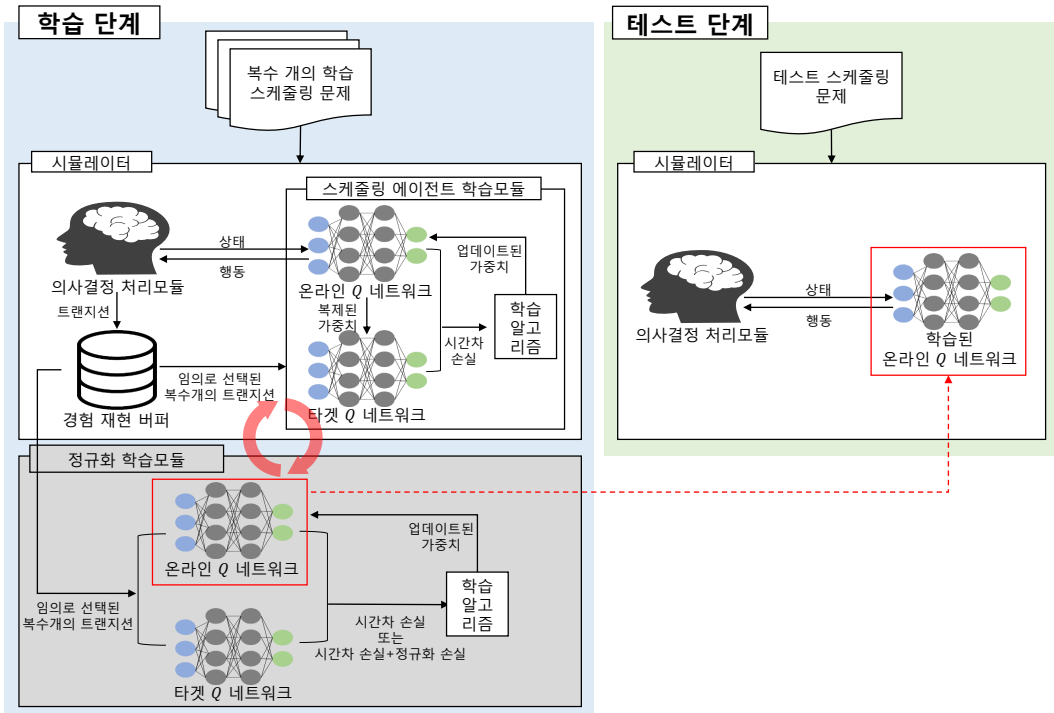

Figure 4.1: 정규화 학습이 포함된 강화학습 기반 스케줄링 구조도

4.2 정규화 학습 과정

강화학습 기반 스케줄링 에이전트의 강건성 확보를 위한 정규화 학습과정은 크게 4 단계로 구성된다. 각 단계를 언급하기에 앞서, 본 연구에서 사용하는 심층 Q 네트워크를 피처층과 Q층으로 구분하여 설명한다. 이에 대한 설명은 아래 그림 4.2에 나타내었다. 피처층은 입력층부터 은닉층까지의 모든 층을 합쳐서 일컬으며, Q층은 마지막 출력층 부분만을 의미한다.

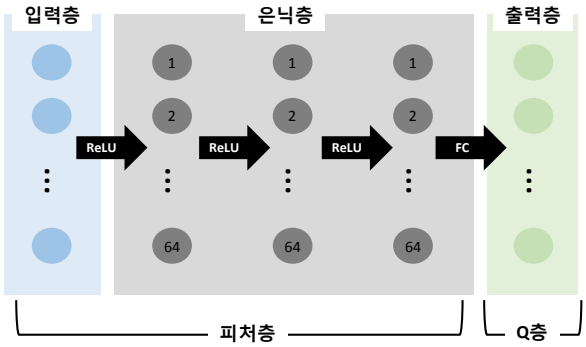

Figure 4.2: 네트워크 피처층과 Q층 구분

본 연구에서 학습할 네트워크의 구조는 아래 그림 4.5과 같다. 입력층부터 은닉층까지의 피처층은 하나의 공통된 네트워크를 활용하며, 학습되는 문제의 수 만큼 최종 출력층인 Q층이 생성된다. 하나의 공통된 피처층이 여러 문제들의 공통적인 지식을 학습하고, 문제별로 나누어진 Q층을 문제별로 따로 학습하면서 문제 별 특성을 학습하도록 유도하였다. 제시된 구조는 모티브로 한 [48]에서 제시된 네트워크 구조를 차용하였다. 그림의 괄호안에 표현된 기호는 각 층의 파라미터인 가중치를 표현하였다. 정규화가 포함된 강화학습 기반 스케줄링 에이전트 학습 과정은 아래 그림 4.4의 알고리즘에 따라 진행된다. 첫 번째로 복수 개의 학습 스케줄링 문제를 한번씩 번갈아 가며 그림 3.4의 알고리즘에 따라 피처층과 Q층을 합친 전체 Q 네트워크를 학습한다.

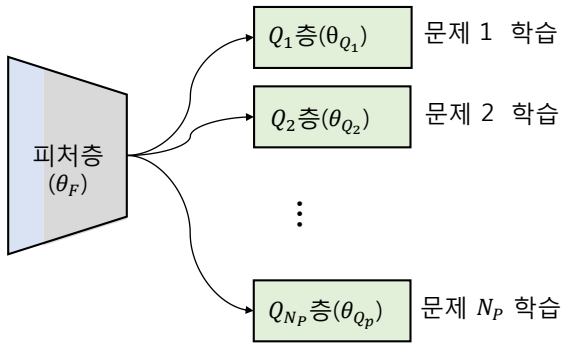

Figure 4.3: 학습 네트워크 구조

이 과정이 끝나면, 복수 개의 스케줄링 문제 중 서로 다른 두 개의 문제($a, b \in P$)를 추출한다. 첫 번째 과정을 통해 학습된 Q 네트워크를 a의 데이터를 이용하여 Q층만 $N_{Q-update}$만큼 학습한다. 마지막으로, b의 데이터를 이용하여 정규화의 가중치인 ϕ를 학습한다. 위 가정을 총 N_S만큼 반복한다. 정규화 가중치까지 학습이 완료되면, ϕ를 고정하고 새로운 하나의 Q층을 초기화하여 모든 스케줄링 문제의 데이터에 대해 학습한다. 정규화 학습에 사용되는 손실함수는 시간차 손실과 정규화 손실이 합쳐진 아래 수식 4.1을 이용한다. 여기서 ϕ는 본 연구에서 학습하고자는 정규화의 가중치이며, 정규화는 $L2$정규화를 이용하고 마지막 층인 Q층에만 적용한다. θ는 전체 Q 네트워크의 파라미터를 의미하고, 앞서 네트워크를 나눈 기준으로 피처층의 파라미터인 θ_F와 Q층의 파라미터인 θ_Q를 합친 파라미터 집합을 의미한다. λ는 정규화 상수이다.

$$L_{reg}(\theta) = \frac{1}{2}\sum_{j=1}^{N_{mini}} \left(y_j - Q(s_j, a_j; \theta)\right)^2 + \frac{1}{2}\lambda\phi\|\theta_Q\|_2 \tag{4.1}$$

Algorithm 3 Training deep Q network agent with regularization for scheduling

Input: P : Multiple scheduling problems set for training,
N_P: number of scheduling problems,
N_S: Iterations of solving scheduling problem,
$N_{Q-update}$: Iterations of Q layer update,
N_{newQ}: Iterations of training new Q layer
Output: Q network with weighted regularization

Initialize : Initialize N_P replay buffers to capacity N_{RB},
Initialize N_P online Q networks with random weights θ,
Initialize N_P target Q networks with random weights $\theta^{-} = \theta$
for *iteration* $\in$ (1,..,N_S) **do**
  for scheduling problem $p \in P$ **do**
    Train Q network of problem p according to algorithm 3.4 line 3-28 using scheduling problem p
  end for
  Choose $a, b \in P$ randomly such that $a \neq b$
  for $iteration_{Qupdate} \in$ (1,..,$N_{Q-update}$) **do**
    Sample random mini-batch of transitions from replay buffer of a
    Calculate loss L using (3.2) and (4.1)
    Perform a gradient descent step on L w.r.t θ_{Q_a}
  end for
  Sample random mini-batch of transitions from replay buffer of b
  Calculate loss L using (3.2) and (4.1)
  Perform a gradient descent step on L w.r.t ϕ of regularization
end for
Drop all existing Q layers and Initialize single Q layer
for $iteration_{newQ} \in$ (1,..,N_{newQ}) **do**
  for scheduling problem $p \in P$ **do**
    Sample random mini-batch of transitions from replay buffer of p
    Calculate loss L from (3.2) and (4.1)
    Perform a gradient descent step on L w.r.t θ_Q
  end for
end for
return Q network with weighted regularization

Figure 4.4: 정규화가 포함된 심층 Q 네트워크 학습

4.2.1 심층 Q 네트워크 학습

본 연구에서 제안하는 방법의 첫 번째 단계로 그림 3.4의 알고리즘에서 제시한 과정을 복수 개의 학습 스케줄링 문제에 반복한다. 네트워크의 구성은 피처층과 Q층으로 나누어져 있으며, Q층은 학습 스케줄링 문제의 수만큼 생성한다. 복수 개의 학습 문제를 순차적으로 하나씩 시뮬레이터로 스케줄링 문제를 풀면서 학습을 진행하며, 선택된 하나의 스케줄링 문제에 해당하는 Q층과 공통된 피처층의 학습을 진행한다. 이 과정을 통해 공통된 피처층은 복수 개 문제 스케줄링의 일반적이고 공통적인 특성을 학습한다. 학습 문제 별로 구분되어 있는 Q층은 서로 다른 문제들에 특화되어 학습되며, 구분된 Q층을 통해 문제 별로 다른 특성을 학습할 수 있다. 아래 그림 4.5은 전체 네트워크가 학습되는 것을 도식화하였다. 그림에서 빗금친 부분은 해당 과정에서 학습되는 층을 의미한다. 첫 번째 과정에서는 정규화를 이용하지 않으며, 시간차 손실을 역전파하여 학습을 진행한다. 적용되는 시간차 손실은 수식 3.3이다.

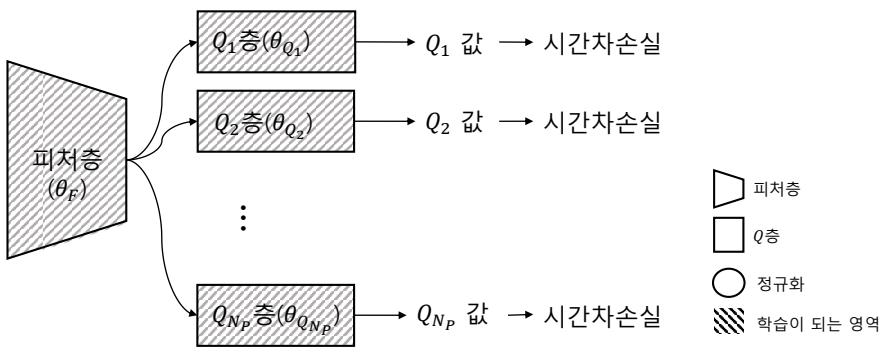

Figure 4.5: 1단계 : 심층 Q 네트워크 학습

4.2.2 Q층 학습

위에서 설명한 첫 번째 단계에서, 복수 개의 문제를 한번씩 스케줄링을 풀면서 심층 Q 네트워크 전체를 학습 한 이후에 시뮬레이터와는 독립적으로 학습을 진행한다. 이번 단계에서는 위의 그림 4.4의 알고리즘 줄6에서 언급한 대로 복수 개의 학습 문제 중 임의의 2개를 추출한다. 그 중 하나를 선택하여(a문제) 해당 문제의 학습 데이터를 경험 재현 버퍼에서 추출하여 학습을 진행한다. 이번 단계에서는 정규화가 추가되어 학습되는 손실함수에 정규화 손실이 추가되며, 학습은 복수 개의 Q층 중 해당 문제에 해당하는 Q층만 학습한다. 이 과정을 $N_{Q-update}$번 반복하고, 이를 통해 선택된 a 문제의 스케줄링 에이전트가 특정 정규화 가중치에 대해 적응하도록 Q층을 파인튜닝(fine-tuning)한다. 아래는 Q층만 학습하는 과정을 도식화하였다. 아래의 그림 4.6 예시에서는 선택된 하나의 문제(a)가 2번 문제라고 가정하고, 학습은 2번 문제에 해당하는 두 번째 Q층만 학습한다. 정규화는 Q층의 파라미터인 가중치에 대한 정규화를 적용하므로 Q층 뒤에 위치하였다. 적용되는 수식은 시간차 손실과 정규화 손실이 합쳐진 수식 4.1을 이용한다.

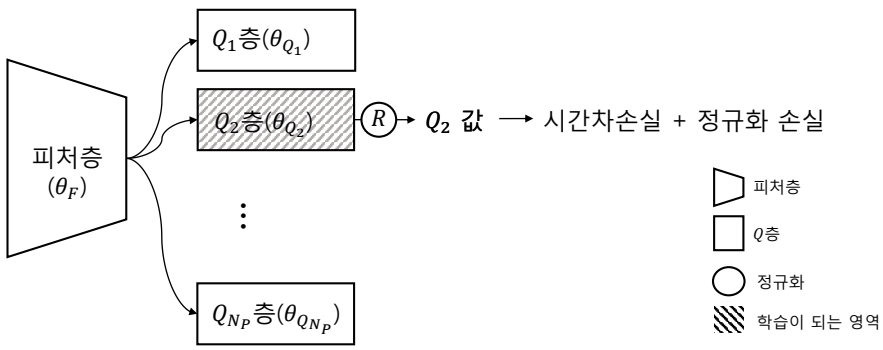

Figure 4.6: 2단계 : 단일 Q층 학습

4.2.3 정규화 가중치 학습

세 번째 단계에서는 정규화의 가중치인 ϕ를 학습한다. 두 번째 단계에서 $N_{Q-update}$ 회 반복해서 특정 학습 문제인 a에 해당하는 Q층을 학습하였고, 동일한 Q층에 대해 다른 학습 문제인 b의 학습 데이터를 경험 재현 버퍼에서 추출하여 정규화 가중치를 학습한다. 일정 횟수 동안 a문제에 적합하도록 학습한 네트워크를 새로운 문제인 b문제의 학습 데이터 대해 손실을 줄이는 방향을 학습함으로써 새로운 문제에서의 모델의 강건함을 얻을 수 있는 방향으로 정규화가 학습된다. 이 과정에서는 한번의 기울기 업데이트를 수행한다. 두 번째 단계와 마찬가지로 손실함수는 시간차손실과 정규화손실을 합친 손실함수인 4.1를 이용한다. 두 번째 단계와 세 번째 단계는 시뮬레이션을 통해 스케줄링을 푸는 횟수인 N_S만큼 반복된다. 반복 과정에서 임의 추출을 통해 학습 문제 a와 b가 바뀌면 정규화 가중치가 서로 다른 문제에서 학습되어 여러 학습 문제에 일반화된 정규화 가중치를 얻을 수 있다. 아래 그림 4.7는 정규화 가중치 학습을 도식화하였다. 두 번째 단계의 예시로 이어서 설명하면, 두 번째 단계에서 2번 문제(a)로 학습된 심층 Q 네트워크에 새로운 문제인 1번 문제(b)를 2번 문제에 해당하는 심층 Q 네트워크를 통과시켜 Q값을 얻는다. 얻은 Q값을 통해 손실함수를 계산하여 그 기울기 업데이트를 정규화의 가중치 ϕ를 업데이트 하는데 사용한다.

4.2.4 새로운 Q층 학습

마지막 단계는 위의 모든 과정이 끝나고, 최종적으로 정규화 가중치를 학습을 완료한 후 진행되는 단계로, 기존에 존재하던 N_P개의 Q층을 모두 제거하고, 새로운 하나의 Q층으로 초기화 시킨다. 이전 단계에서 학습한 피처층과 정규화 가중치는 고정한 채, 새로운 하나의 Q층에 대해 N_P개의 학습 스케줄링 문제를 N_{newQ}만큼 반복해서 학습한다. 어느정도 일반화된 정규화 가중치를 통해 복수 개의 문제를 다시 학습하면서 하나의 Q

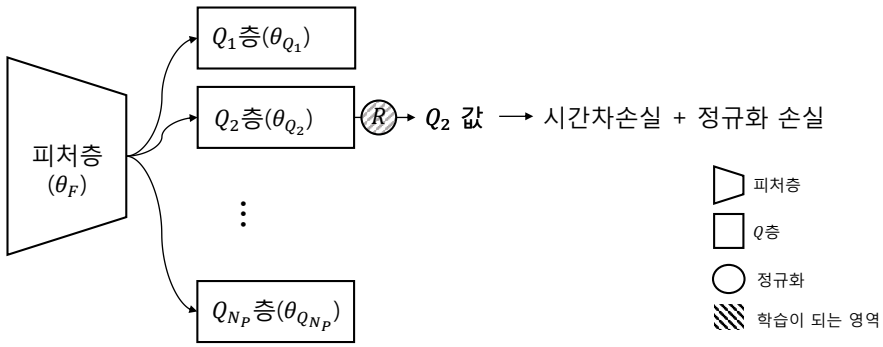

Figure 4.7: 3단계 : 정규화 가중치 학습

층을 최종적으로 학습하고, 이를 최종 모델로 테스트에 사용하게 된다. 최종적인 모델의 형태는 아래의 그림 4.8과 같다.

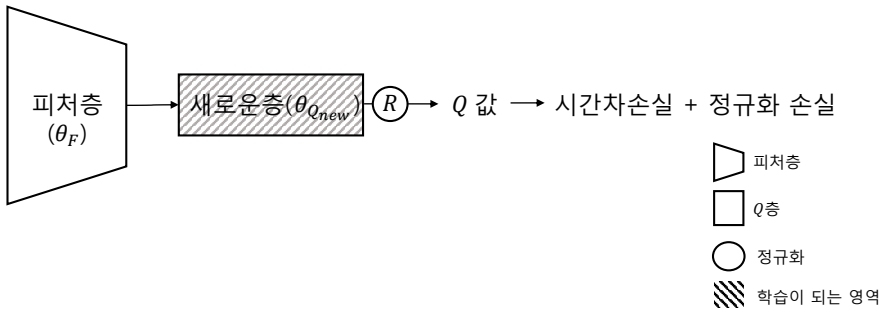

Figure 4.8: 4단계 : 새로운 Q층 학습

제 5 장 실험 결과

5.1 데이터셋

본 연구에서 제안하는 정규화 학습 기법의 성능을 검증하기 위해 표 5.3, 표 5.4, 표 5.5와 같이 총 3가지의 데이터셋을 구성하여 성능을 검증한다. 본 연구에서 제안한 강화학습의 상태, 행동과 네트워크 구조는 잡 타입 개수에 대해 강건하지 못한 구조이므로, 잡 타입 개수를 다르게하여 스케줄링 문제의 복잡도가 서로 다른 3가지의 데이터를 구성하였다. 실험에 사용된 각 데이터셋의 구성은 아래의 표 5.1에 설명되어 있으며, 각 잡 타입은 아래의 표 5.2에 정의하였다. 본 연구에서는 각 작업공정의 설비의 수도 변화할 수 있는 요소이므로, 대체 가능한 설비의 집합인 $A(O_{i,j})$는 DA공정은 최대 15대이하, WB공정은 최대 25대이하로 구성된다. 각 데이터셋은 구성하는 잡 타입의 종류와 수가 차이가 있다. 공통적으로 총 설비수는 35대, 작업 소요시간은 DA의 경우 1.6시간, WB는 4.8시간이며, 셋업 교체 시간 DA은 1시간, WB는 3시간으로 모두 동일하다.

Table 5.1: 데이터셋 별 구성

데이터셋 번호	구성 잡 타입
1	$[J_1,J_3,J_4,J_6]$
2	$[J_1,J_2,J_3,J_4,J_6,J_7]$
3	$[J_1,J_2,J_3,J_4,J_5,J_6,J_7,J_8]$

아래의 각 데이터셋을 설명한 표 5.3, 표 5.4, 표 5.5에서 잡 타입 별 생산요구량과 설비 초기 셋업에 관한 정보는 숫자 벡터형태로 표현하였으며, 이는 각 데이터셋을 구성하는 잡 타입에 해당하는 숫자이다. 예를 들어, 데이터셋1에 해당하는 것은 대괄호안에 나타내는 숫자는 각각 데이터셋1을 구성하는 J_1,J_3,J_4,J_6 순서 표현된다.

각 데이터셋은 총 8개의 스케줄링 문제로 구성되어있다. 우선 하나의 기준 문제를 구성하고, 본 연구에서 가정한 생산환경의 변화인 작업공정 별 설비 비율 변화, 설비 초기 셋업 변화, 잡 타입별 생산 비율 변화를 각 기준 문제에 대해 적용하였다. 기준 문제는 각 데이터셋의 1번 문제에 해당하며, 생산환경을 하나씩 변화시킨 데이터셋은 2,3,4에 순서대로 해당한다. 5번 문제는 세 가지의 변동 요소가 모두 적용된 데이터셋이며, 6번 문제는 5번 문제 대비 변동의 폭을 크게 설정한 문제이다. 특히, 설비 초기 셋업이 특정 잡 타입에 대부분 할당된 극단적인 문제이다. 7번 문제는 5번 문제처럼 세 가지의 변동 요소가 모두 적용되면서 생산요구량을 기준 문제 대비 약 1.5배 증가한 문제이다. 8번 문제의 경우 기준문제 대비 생산요구량을 각 잡 타입 별로 2배씩 한 문제이다. 7,8번 문제는 현재 실험이 실제 공장에서 다루는 규모보다 훨씬 작은 규모의 문제이므로, 스케일업의 관점에서 성능을 확인하기 위한 문제이다. 각 데이터셋의 특징에 대한 설명은 아래의 표에 설명하였다. 생산요구량에 해당하는 단위는 랏 단위로 표기하였다.

Table 5.2: 잡 타입 설명

J_i	$O_{i,j}$	작업공정	$A(O_{i,j})$
J_1	$O_{1,1}$	DA	M_1 - M_{15}
	$O_{1,2}$	WB	M_{16} - M_{40}
J_2	$O_{2,1}$	DA	M_1 - M_{15}
	$O_{2,2}$	WB	M_{16} - M_{40}
J_3	$O_{3,1}$	DA	M_1 - M_{15}
	$O_{3,2}$	WB	M_{16} - M_{40}
	$O_{3,3}$	DA	M_1 - M_{15}
	$O_{3,4}$	WB	M_{16} - M_{40}
J_4	$O_{4,1}$	DA	M_1 - M_{15}
	$O_{4,2}$	WB	M_{16} - M_{40}
	$O_{4,3}$	DA	M_1 - M_{15}
	$O_{4,4}$	WB	M_{16} - M_{40}
J_5	$O_{5,1}$	DA	M_1 - M_{15}
	$O_{5,2}$	WB	M_{16} - M_{40}
	$O_{5,3}$	DA	M_1 - M_{15}
	$O_{5,4}$	WB	M_{16} - M_{40}
J_6	$O_{6,1}$	DA	M_1 - M_{15}
	$O_{6,2}$	WB	M_{16} - M_{40}
	$O_{6,3}$	DA	M_1 - M_{15}
	$O_{6,4}$	WB	M_{16} - M_{40}
	$O_{6,5}$	DA	M_1 - M_{15}
	$O_{6,6}$	WB	M_{16} - M_{40}
J_7	$O_{7,1}$	DA	M_1 - M_{15}
	$O_{7,2}$	WB	M_{16} - M_{40}
	$O_{7,3}$	DA	M_1 - M_{15}
	$O_{7,4}$	WB	M_{16} - M_{40}
	$O_{7,5}$	DA	M_1 - M_{15}
	$O_{7,6}$	WB	M_{16} - M_{40}
J_8	$O_{8,1}$	DA	M_1 - M_{15}
	$O_{8,2}$	WB	M_{16} - M_{40}
	$O_{8,3}$	DA	M_1 - M_{15}
	$O_{8,4}$	WB	M_{16} - M_{40}
	$O_{8,5}$	DA	M_1 - M_{15}
	$O_{8,6}$	WB	M_{16} - M_{40}
	$O_{8,7}$	DA	M_1 - M_{15}
	$O_{8,8}$	WB	M_{16} - M_{40}

Table 5.3: 데이터셋1의 스케줄링 문제 설명

문제 번호	총 생산 요구량(개)	생산 환경			문제 특징
		잡 타입별 생산요구량(개)	공정 별 설비 비율(대)	설비 초기 셋업(개)	
1-1	65	[15,15,20,15]	DA:10 WB:25	DA:[2,1,1,6] WB:[3,5,5,12]	기준문제
1-2	65	[15,15,20,15]	DA:12 WB:23	DA:[2,1,2,7] WB:[1,5,5,12]	공정 별 설비 비율 변화
1-3	65	[15,15,20,15]	DA:10 WB:25	DA:[4,3,3,0] WB:[1,7,8,9]	설비 초기 셋업 변화
1-4	65	[10,20,25,10]	DA:10 WB:25	DA:[2,1,1,6] WB:[3,5,5,12]	잡 타입별 생산요구량 비율 변화
1-5	75	[30,10,10,25]	DA:8 WB:27	DA:[3,1,1,3] WB:[7,7,6,7]	세 가지 생산환경 모두 변화
1-6	65	[5,25,30,5]	DA:15 WB:20	DA:[0,9,6,0] WB:[2,6,5,7]	세 가지 생산환경 극단적 변화
1-7	90	[30,15,15,30]	DA:9 WB:26	DA:[2,1,2,4] WB:[7,6,6,7]	세 가지 생산환경 모두 변화 및 생산요구량 증가
1-8	130	[30,30,40,30]	DA:10 WB:25	DA:[2,1,1,6] WB:[3,5,5,12]	기준문제 대비 생산요구량 증가(2배)

Table 5.4: 데이터셋2의 스케줄링 문제 설명

문제 번호	총 생산 요구량(개)	생산 환경			문제 특징
		잡 타입별 생산요구량(개)	공정 별 설비 비율(대)	설비 초기 셋업(개)	
2-1	60	[10,5,10,15,15,5]	DA:10 WB:25	DA:[1,0,0,3,4,2] WB:[6,1,1,6,11,0]	기준문제
2-2	65	[10,5,10,15,15,5]	DA:12 WB:23	DA:[1,0,0,3,5,3] WB:[3,2,1,6,11,0]	공정 별 설비 비율 변화
2-3	65	[10,5,10,15,15,5]	DA:10 WB:25	DA:[0,0,1,3,4,2] WB:[2,1,1,8,13,0]	설비 초기 셋업 변화
2-4	60	[10,10,10,20,5,5]	DA:10 WB:25	DA:[1,0,0,3,4,2] WB:[6,1,1,6,11,0]	잡 타입별 생산요구량 비율 변화
2-5	65	[15,15,5,10,10,10]	DA:8 WB:27	DA:[1,1,1,2,3,0] WB:[2,2,2,4,10,7]	세 가지 생산환경 모두 변화
2-6	60	[5,5,20,20,5,5]	DA:15 WB:20	DA:[8,0,0,0,7,0] WB:[2,0,1,6,11,0]	세 가지 생산환경 극단적 변화
2-7	90	[10,10,15,20,20,15]	DA:9 WB:26	DA:[4,2,1,1,1,0] WB:[6,2,1,6,11,0]	세 가지 생산환경 모두 변화 및 생산요구량 증가
2-8	120	[20,10,20,30,30,10]	DA:10 WB:25	DA:[1,0,0,3,4,2] WB:[6,1,1,6,11,0]	기준문제 대비 생산요구량 증가(2배)

Table 5.5: 데이터셋3의 스케줄링 문제 설명

문제 번호	총 생산 요구량(개)	생산 환경			문제 특징
		잡 타입별 생산요구량(개)	공정 별 설비 비율(대)	설비 초기 셋업(개)	
3-1	85	[10,5,10,15,15,15,15,5,10]	DA:10 WB:25	DA:[1,0,1,1,1,2,2,2] WB:[5,2,3,3,2,3,3,4]	기준문제
3-2	85	[10,5,10,15,15,15,15,5,10]	DA:12 WB:23	DA:[1,2,1,1,1,2,2,2] WB:[5,0,3,3,2,3,3,4]	공정 별 설비 비율 변화
3-3	85	[10,5,10,15,15,15,15,5,10]	DA:10 WB:25	DA:[3,0,1,2,1,3,0,0] WB:[2,1,4,4,2,5,3,4]	설비 초기 셋업 변화
3-4	95	[15,20,5,20,10,10,10,5]	DA:10 WB:25	DA:[1,0,1,1,1,2,2,2] WB:[5,2,3,3,2,3,3,4]	잡 타입별 생산요구량 비율 변화
3-5	85	[10,5,5,10,10,20,10,15]	DA:8 WB:27	DA:[2,1,1,1,1,0,1,1] WB:[1,0,6,3,2,6,3,6]	세 가지 생산환경 모두 변화
3-6	85	[20,20,10,10,10,5,5,5]	DA:15 WB:20	DA:[0,0,0,0,0,5,6,4] WB:[1,0,3,3,2,4,3,4]	세 가지 생산환경 극단적 변화
3-7	120	[15,10,10,15,10,20,20,20]	DA:9 WB:26	DA:[3,0,1,1,1,2,1,0] WB:[0,1,5,2,2,6,3,7]	세 가지 생산환경 모두 변화 및 생산요구량 증가
3-8	170	[20,10,20,30,30,30,30,10,20]	DA:10 WB:25	DA:[1,0,1,1,1,2,2,2] WB:[5,2,3,3,2,3,3,4]	기준문제 대비 생산요구량 증가(2배)

5.2 실험 과정

실험은 각 데이터셋에서 학습할 스케줄링 문제와 테스트할 스케줄링 문제를 구분하여 진행한다. 각 데이터셋에서 1-4번 문제를 제안기법의 학습 스케줄링 문제로 선택한다. 본 연구에서 생산환경 변화를 3가지 요소로 제한하였기 때문에, 각 요소를 변화시킨 문제를 학습하여 여러 생산환경의 특성을 학습할 수 있도록한다. 나머지 4개의 다양한 변동을 가정한 스케줄링 문제에 대해 테스트를 진행한다.

제안 기법을 통해 학습된 모델의 성능은 룰 기반의 스케줄링과 다양한 방식으로 학습된 강화학습 기반 스케줄링 에이전트와 비교함으로써 검증한다. 성능 비교를 위해 사용될 각 스케줄링 방법은 아래의 표 5.6에 정리하였다.

Table 5.6: 성능 비교 기법 정리

스케줄링 방법	방법 명칭	설명
룰 기반 스케줄링 (Rule)	First In First Out(FIFO)	가장 먼저 도착하는 잡 선택
	Shortest Setup Time(SSU)	셋업 교체 시간이 제일 적은 잡 선택
	Most Operation Remaining(MOR)	남은 작업이 가장 많이 남은 잡 선택
강화학습 기반 스케줄링(단일문제) (RL-single)	1번 문제로 학습(single1)	각 데이터셋의 1번 문제로만 학습
	2번 문제로 학습(single2)	각 데이터셋의 2번 문제로만 학습
	3번 문제로 학습(single3)	각 데이터셋의 3번 문제로만 학습
	4번 문제로 학습(single4)	각 데이터셋의 4번 문제로만 학습
강화학습 기반 스케줄링(복수문제) (RL-multiple)	임의추출모델(sampling)	임의로 학습문제를 추출하여 반복학습
	정규화가 포함된 임의추출모델(sampling w/ reg.)	$L2$정규화 포함하여 임의로 학습문제를 추출하여 반복학습
	순차추출모델(sequential)	순차로 학습문제를 추출하여 반복학습
	정규화가 포함된 순차추출모델(sequential w/ reg.)	$L2$정규화 포함하여 순차로 학습문제를 추출하여 반복학습

단일 문제 학습은 학습용 스케줄링 문제 4개를 각각 한 문제씩 학습한 강화학습 모델을 의미한다. 복수 문제 학습은 제안기법과 마찬가지로 학습용 스케줄링 문제 4개를 모두 학습한 모델이다. 복수 문제 학습 중 임의 추출 모델은 학습용 스케줄링 문제 4개 중 하나씩 임의로 추출하여 문제를 학습하고 추출과정을 반복하는 모델이며, 순차 추출 모델은 1,2,3,4번 문제 순서대로 한 문제씩 추출하여 문제를 학습하고 추출과정을

반복하는 모델이다. 제안기법과 정규화가 사용된 모델은 모두 $L2$ 정규화를 이용하였다. 강화학습 기반의 모델들은 모두 3000번(N_S)의 스케줄링 문제를 풀면서 학습하도록 하였다. 단일 문제 학습 모델의 경우 학습과정에서 가장 좋은 성능을 보이는 모델을 선택하였다. 복수 문제 학습 모델의 경우 3000번 스케줄링 학습 과정 후 최종 모델을 기준으로 성능비교를 하였다. 3가지의 룰 기반의 스케줄링 방법 중 가장 좋은 성능의 룰과 비교하였다. 제시한 3가지 룰 기반 스케줄링은 [49]를 참고하였다. 또한 테스트 문제에 대한 성능이 어느정도인지 판단하기 위해, 각 테스트 문제를 단일 문제로 학습한 강화학습 스케줄링 에이전트의 성능 또한 제시하였다. 이는 성능 비교 그래프에서 desired performance로 제시한다.

5.3 실험 세팅

5.3.1 강화학습 실험 세팅

심층 Q 네트워크를 학습하기 위한 옵티마이저(optimizer)로 Adam [50]을 이용하였다. 또한 심층 Q 네트워크의 구조는 앞서 설명한대로, 은닉층이 3개로 구성되어 있고 각각은 64개의 노드를 가진다. 입력층과 출력층은 실험한 데이터셋에 따라 달라진다. 강화학습에서 사용된 탐색 방식인 입실론 그리디 방식에서 입실론을 점점 줄여나가는 방식을 활용하였으며, 초기 ϵ은 0.3에서 선형적으로 0.01까지 줄여나가는 방식을 택했다. 또한, 학습의 안정성을 위해 네트워크의 파라미터인 가중치 값이 -1과 1사이로 제한하였다. 강화학습에 필요한 하이퍼파라미터(hyperparameter)는 아래의 표 5.7에 정리하였다. 아래의 표 5.7는 각 문제를 단일 문제로 학습할 때 무작위 탐색을 통해서 성능이 평균적으로 가장 높은 하이퍼파라미터로 선정하였다. 강화학습을 구현하기 위해 구글의 딥러닝 라이브러리인 tensorflow를 이용하였다 [51].

Table 5.7: 강화학습 하이퍼파라미터

하이퍼파라미터	값
Adam 옵티마이저 학습률	10^{-4}
Adam 옵티마이저 입실론	10^{-4}
임의 행동이 선택될 확률(ϵ) 초기값	0.3
임의 행동이 선택될 확률(ϵ) 최종값	0.01
강화학습 할인율, γ	0.7
경험 재현 버퍼 크기, N_{RB}	10^6
미니 배치 크기	64
타겟 Q 네트워크 업데이트 주기, C	10

5.3.2 정규화 학습 실험 세팅

제안 기법은 학습 문제 수에 따라 마지막 Q층의 개수가 변화한다. 본 연구의 실험에서는 4개(N_P)의 학습 스케줄링 문제를 이용하였으므로 총 4개의 Q층으로 구성된다. 정

규화 학습 2단계에서 진행되는 Q층만 학습하는 과정은 100회씩 진행된다($N_{Q-update}$). 또한, 마지막으로 새로운 단일 Q층으로 초기화하여 학습하는 과정은 500회 반복한다 (N_{newQ}). 정규화 손실함수에 적용되는 정규화 가중치인 λ는 10^{-4}로 설정하였다. 따로 하이퍼파라미터로 정의하지 않았지만, 3단계에 해당하는 정규화 가중치 학습 단계는 2단계를 학습하는 $N_{Q-update}$ 당 1회씩으로 고정하였다. 이는, 많은 기울기 업데이트를 진행할 경우 경험적으로 정규화 가중치가 큰 음수값으로 치우쳐 학습에 악영향을 주기 때문이다. 즉, ϕ에 대한 기울기 업데이트를 조절하면서 모델의 성능을 실험적으로 비교하여 업데이트 횟수를 설정하였다. 또한, 정규과 가중치인 ϕ의 영역을 -1과 1사이로 제한하여 학습이 안정적으로 될 수 있도록 하였다.

5.4 실험 결과

실험 결과는 표 5.6에 제시된 방법들과 본 연구에서 제안한 방법의 테스트 성능을 비교한다. 그 결과는 아래의 그림 5.1, 그림 5.2, 그림 5.3에 나타내었다. 각 그림은 순서대로 데이터셋1,2,3에 해당하는 결과를 의미한다. 각 그림안에는 4개의 바차트로 구성되어 있으며, 위에서부터 차례대로 테스트 문제 5,6,7,8에 대한 결과이다. 순서대로 (a),(b),(c),(d)라 표현하였다. 그림의 x축에 표현된 명칭은 모두 표 5.6의 방법 명칭에서의 영어 표현을 이용하였고, 룰 기반의 경우 세 가지 룰 기반 스케줄링 중 가장 성능이 높은 룰 하나만을 선택하여 표시하였다. 제안된 기법은 Proposed로 표현하였다. 그리고, 스케줄링 방법에 따라 명암을 주어 구분하였다. 또한, 제안 기법의 성능의 정도를 판단하기 위해 해당 테스트 문제를 학습한 모델의 결과를 빨간선으로 제시하였다.

실험 결과 중 2가지 테스트 문제를 제외하고 제안기법이 다른 비교 모델들보다 높은 성능을 보인다. 예외 케이스는 1-5, 3-6번 테스트 문제에서 단일 문제로 학습한 모델보다 좋지 못한 성능이 도출되었다. 1-5번 문제에서는 학습 문제 중 1-1문제를 단일로 학습한 single1이 제안 기법보다 좋은 성능을 보였다. 하지만 single1 모델은 다른 테스트 데이터에 대해서는 제안 기법에 비해 월등히 좋지 못한 성능을 보였다. 즉, 하나의 테스트 문제에 대해서는 좋은 성능을 보일 수 있지만 제안기법이 여러 테스트 문제에 대해서 더욱 강건한 성능을 보임을 알 수 있다. 이를 통해, 본 연구에서 가정한 생산환경의 변화에 다른 모델들에 비해 잘 대응됨을 확인할 수 있고, 이는 제안 기법이 더욱 강건한 강화학습 기반 스케줄링 모델을 학습했다는 것을 보여준다. 또한, 모든 실험 결과에서 (b)에서 (d)로 갈수록 생산요구량이 많아지는데, 스케일이 커지는 문제에서 제안 기법의 성능개선이 두드러짐을 확인 할 수 있다. 이러한 현상은 잡 타입을 더 많이 포함하는 데이터셋2, 데이터셋3에서도 유사한 경향성을 확인 할 수 있다. 이는 스케일업 관점에서도 제안기법으로 학습된 강화학습 스케줄링 에이전트의 강건함이 유효함을

확인하였다.

제안 기법을 통해 학습된 정규화 가중치 분포를 살펴보면 아래의 그림 5.4과 같다. 가로축은 가중치 값을 의미하고, 세로축은 해당 값의 빈도수를 나타낸다. 각 데이터셋에 따라 정규화의 가중치 총 개수가 다르므로, 빈도수의 차이가 발생한다. 성능을 비교한 강화학습 기반 모델 중 정규화를 활용한 두 가지의 모델은(sampling w/ reg., sequential w/ reg.) 모두 가중치가 1인 $L2$ 정규화를 이용하였다. 제안 기법에서는 정규화의 가중치를 ϕ로 두고 이를 학습함으로써 기존 모델 대비 성능 개선을 이뤄냈다. 학습된 형태를 살펴보면, 초기 정규화 가중치는 모두 -10^{-3} 에서 10^{-3}로 초기화된 상태에서 학습을 진행하였고, 대체적으로 음의 값을 갖는 분포로 학습되었다. 데이터셋1과 2를 학습한 정규화 가중치는 비슷한 분포 형태로 학습이 되었다. 대부분 0 근방의 수치를 가짐을 알 수 있으며 값의 분포로 봤을 때 -0.1이 안되는 수치까지 가중치의 값이 퍼져있는 것을 확인 할 수 있다. 반면, 데이터셋3에서는 가중치의 분포가 좀 더 음수쪽으로 치우쳐져있으며, -1에서 빈도가 높은 것을 확인 할 수 있다. 이는 학습 당시에 모든 가중치값을 -1과 1사이로 제한하였기 때문이다. 가중치값들이 이렇게 음수로 가지는 이유는 손실함수에 정규화 손실을 더하였고, 손실함수를 최소화하는 방향으로 학습하기 때문에 자연스럽게 가중치들이 음수를 갖게 된 것으로 예상된다. 이를 통해, 정규화의 가중치가 1로 고정된 모델보다 학습을 통해 다양한 값의 가중치를 가질 때 스케줄링 모델의 성능이 개선되는 것을 확인 할 수 있다.

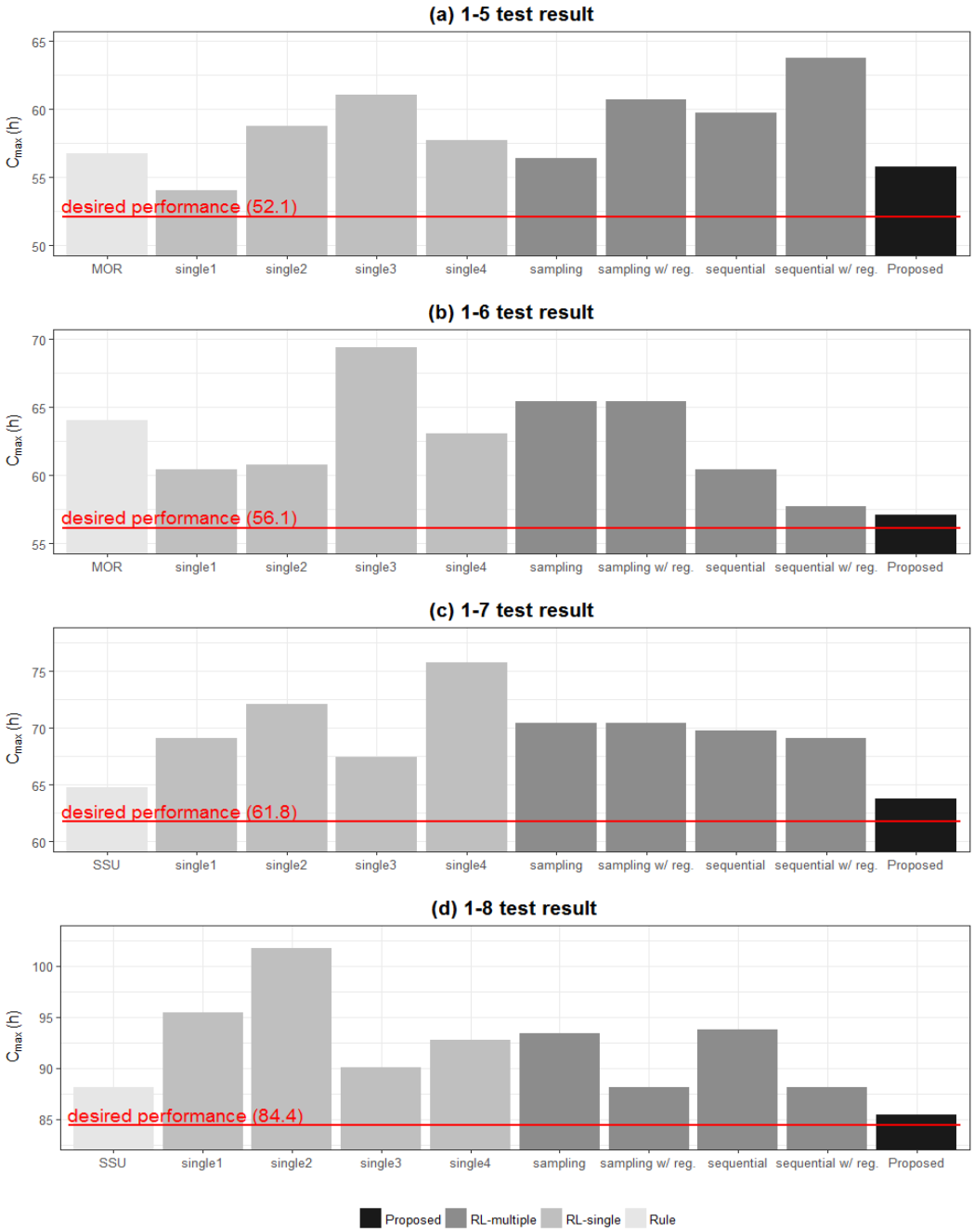

Figure 5.1: 데이터셋1 테스트 결과

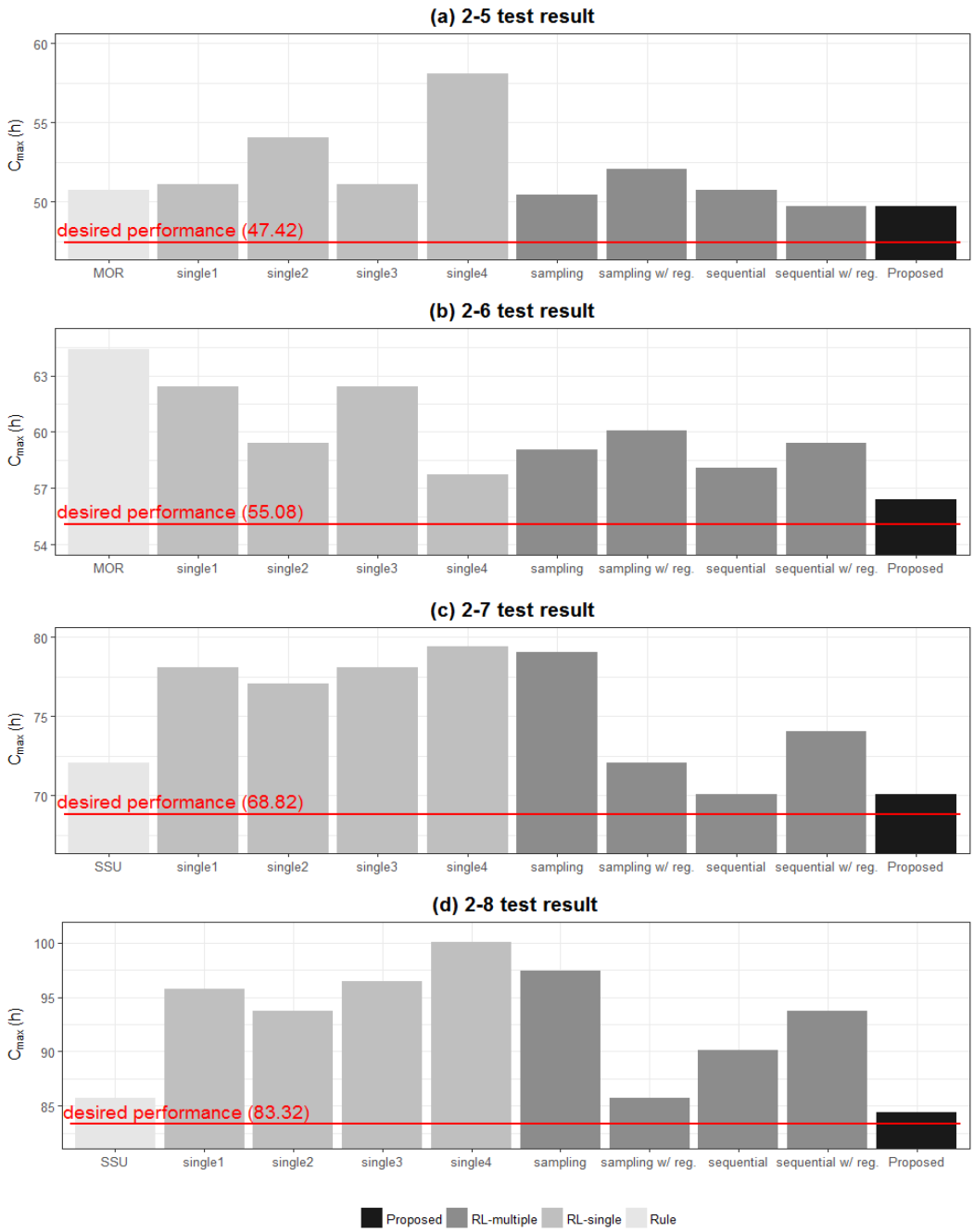

Figure 5.2: 데이터셋2 테스트 결과

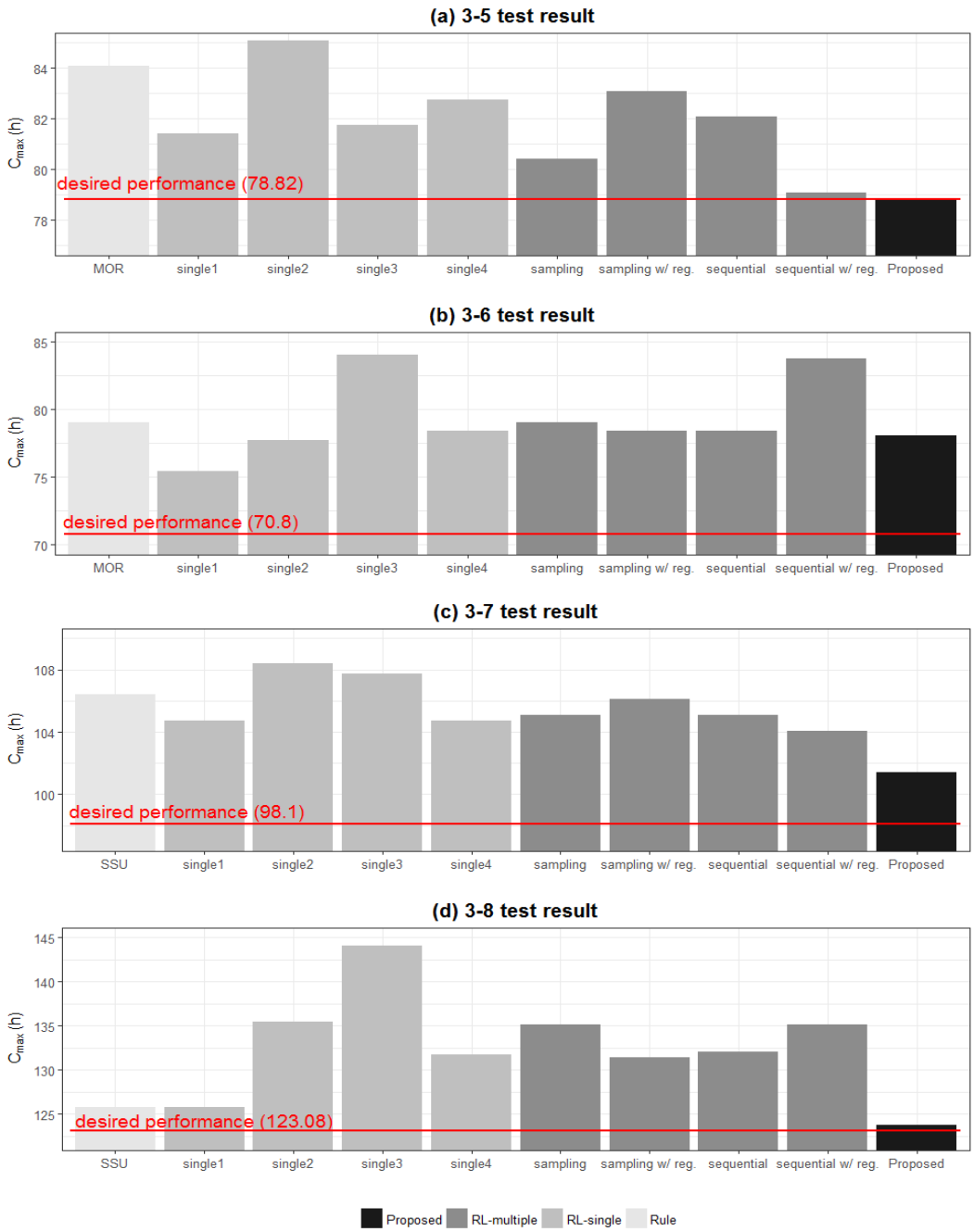

Figure 5.3: 데이터셋3 테스트 결과

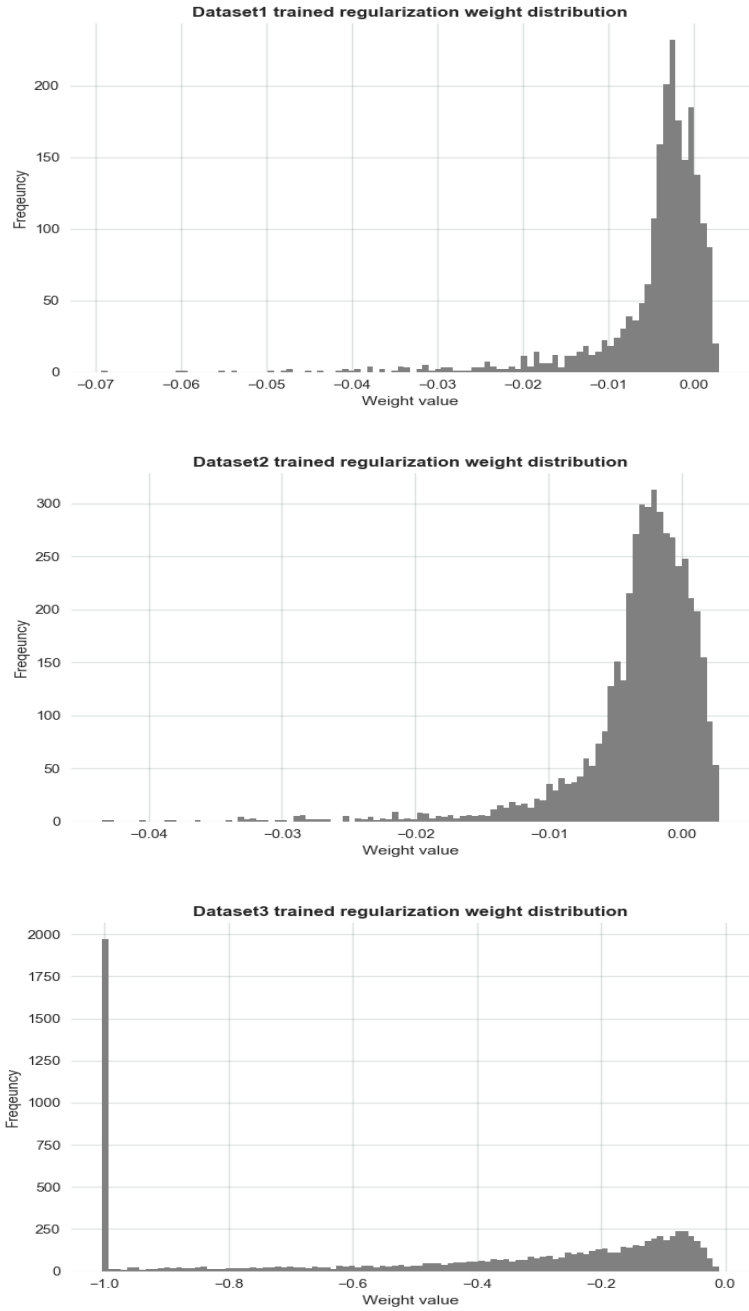

Figure 5.4: 학습된 정규화 가중치 히스토그램

제 6 장 결론

6.1 결론

반도체 패키징 라인에서의 다중 칩 제품 생산은 반복적인 재유입과 셋업 교체 시간의 소요로 인해 스케줄링의 복잡도가 높다. 또한, 반도체 패키징 라인은 시장 수요의 높은 변동성과 설비 유지 보수 등에 의해 생산환경이 빈번히 변화하는 특성을 지니고 있다. 복잡한 스케줄링 문제를 해결하기 위해 전역 최적화를 목표로 하는 심층 강화학습이 스케줄링 연구에 적용되고 있으며, 연구의 활용성 측면에서 빈번한 생산환경 변화에 성능의 큰 저하없이 빠르게 대응할 수 있는 심층 강화학습 기반 스케줄링 연구의 필요성이 대두되고있다.

본 연구에서는 빈번한 생산환경 변화에 재학습 없이 대응할 수 있는 강건한 심층 강화학습 기반 반도체 패키징 라인 스케줄링 모델 학습을 위해 정규화 가중치 학습에 대한 연구를 수행하였다. 반도체 패키징 라인 스케줄링의 목적함수로 전체공기을 이용하였으며, 강화학습을 적용하기 위해 문제의 특성에 맞게 상태, 행동, 보상을 정의하였다. 심층 Q 네트워크를 이용하여 스케줄링을 위한 강화학습 모델을 학습하였으며, 강건성 확보를 위한 정규화 학습을 위해 크게 4단계로 구분하여 학습을 진행하였다. 각 단계에서는 복수 개의 문제의 일반성과 각 문제의 특수성을 학습 할 수 있도록 설계하였다.

제안 기법으로 학습된 반도체 패키징 라인 스케줄링 모델이 잡 타입 별 생산 비율 변화, 설비 초기 셋업 상태 변화, 작업공정 별 설비 비율 변화에 해당하는 생산환경 변화에 대해서 강건한 성능을 보이는지 실험을 통해 확인하였다. 복잡도가 서로 다른 데이터셋에서의 테스트를 통해 성능을 검증하였으며, 룰 기반의 스케줄링 방법과 다양

한 강화학습 기반의 스케줄링 방법들과 비교하여 제안 기법으로 학습된 모델의 성능이 우수함을 확인하였다. 또한 생산요구량이 증가하는 스케일업 된 스케줄링 문제에서도 강건함을 확인했으며, 스케일이 커질수록 다른 방법과의 성능 차이가 더욱 커짐을 확인할 수 있었다.

본 연구에서 제안한 기법이 강건한 반도체 패키징 라인 스케줄링 모델을 학습하는데 유효함을 실험을 통해 확인하였고, 이는 실제 공장에서의 강화학습 기반 스케줄링 모델의 활용도를 높이는 연구 결과이다. 또한, 스케일업 된 스케줄링 문제에서의 성능의 우수함은 큰 크기의 스케줄링 문제를 작은 크기의 스케줄링 문제로 축소하여 강화학습 기반 스케줄링 모델을 빠르게 학습하여 적용할 수 있다는 근거로서 활용 될 수 있다.

6.2 한계점 및 향후 연구

본 연구에서는 생산환경이 변화할 때, 학습하지 않은 새로운 문제에 대한 강화학습 기반 스케줄링 에이전트의 성능을 높이고자 정규화의 가중치를 학습하는 방식으로 강화학습 모델의 강건성을 확보하고자 하였다. 그러나, 본 연구에서 제안한 방법에서도 크게 두 가지의 한계점을 지니고 있으며, 추후 연구를 통해 더욱 강건한 강화학습 기반 스케줄링 에이전트 학습이 가능할 것으로 예상된다.

먼저, 여기서 가정한 생산환경의 변화는 크게 잡 타입 별 생산비율의 변화, 설비 초기 셋업 상태 변화, 작업공정 별 설비 비율 변화로 제한하였다. 세 가지 이외의 많은 변동요인이 있을 수 있으며, 대표적으로 잡 타입 수 변화를 고려하지 않았다. 잡 타입 수 변화를 고려하지 않았기 때문에, 데이터셋을 잡 타입 수를 변화시키면서 구성하였다. 본 연구에서는 정의한 상태와 행동이 모두 잡 타입 수에 종속적인 형태로 구성하였기 때문에 잡 타입 수 변화에 대응하지 못한다. 하지만 추후 연구에서는 잡 타입 수에 종속적이지 않은 형태의 상태와 행동을 구현함으로써 잡 타입 수 변화에 대해서도 강건한 모델을 만들 수 있을 것이라 예상된다. 또한, 다양한 변동요인이 있을 수 있으므로, 이에 대한 범주를 확장하는 연구가 필요하며, 추후 연구를 통해서 실제 반도체 패키징 라인에서의 강화학습 기반의 모델 활용도가 높아질 것이다.

다음으로, 정규화 학습에 있어서 본 연구에서는 $L2$ 정규화만을 이용하여 실험하였다. 하지만 제안한 기법은 특정 정규화에만 국한되지 않으므로, 사용하지 않은 $L1$ 정규화도 이용할 수 있다. 뿐만아니라, 정규화 자체를 신경망으로 구성하여 신경망의 가중치를 학습하는 형태를 취할 수도 있다. 다양한 형태의 정규화를 제안기법을 통해 학습하고 이를 실험한다면 제안 기법이 유효함을 검증할 수 있을 것이라 예상된다.

참고 문헌

[1] Yong-Hee Han and Jin Young Choi. A gspn-based approach to stacked chips scheduling problem. *IEEE Transactions on Semiconductor Manufacturing*, 23(1):4–12, 2009.

[2] Sang-Jin Lee and Tae-Eog Lee. Scheduling a multi-chip package assembly line with reentrant processes and unrelated parallel machines. In *Proceedings of the 40th Conference on Winter Simulation*, pages 2286–2291. Winter Simulation Conference, 2008.

[3] Mike Tao Zhang, Jeff Fu, and Eric Zu. Dynamic capacity modeling with multiple re-entrant workflows in semiconductor assembly manufacturing. In *IEEE International Conference on Automation Science and Engineering, 2005.*, pages 160–165. IEEE, 2005.

[4] Gerald Weigert, A Klemmt, and S Horn. Design and validation of heuristic algorithms for simulation-based scheduling of a semiconductor backend facility. *International Journal of Production Research*, 47(8):2165–2184, 2009.

[5] Yang Song, Mike Tao Zhang, Jingang Yi, Lawrence Zhang, and Li Zheng. Bottleneck station scheduling in semiconductor assembly and test manufacturing using ant colony optimization. *IEEE Transactions on Automation Science and Engineering*, 4(4):569–578, 2007.

[6] Beom-suk Chung, Junseok Lim, In-Beom Park, Jonghun Park, Minseok Seo, and Jinwook Seo. Setup change scheduling for semiconductor packaging facilities using a genetic algorithm with an operator recommender. *IEEE Transactions on Semiconductor Manufacturing*, 27(3):377–387, 2014.

[7] Fantahun M Defersha and Mingyuan Chen. A parallel genetic algorithm for a flexible job-shop scheduling problem with sequence dependent setups. *The international journal of advanced manufacturing technology*, 49(1-4):263–279, 2010.

[8] R Moghaddas and M Houshm. Job-shop scheduling problem with sequence dependent setup times. 2008.

[9] Peng Qu and Scott J Mason. Metaheuristic scheduling of 300-mm lots containing multiple orders. *IEEE Transactions on Semiconductor Manufacturing*, 18(4):633–643, 2005.

[10] Yi-Chi Wang and John M Usher. Application of reinforcement learning for agent-based production scheduling. *Engineering Applications of Artificial Intelligence*, 18(1):73–82, 2005.

[11] AS Xanthopoulos, Dimitris E Koulouriotis, Vassilios D Tourassis, and Dimitris M Emiris. Intelligent controllers for bi-objective dynamic scheduling on a single machine with sequence-dependent setups. *Applied Soft Computing*, 13(12):4704–4717, 2013.

[12] Zhicong Zhang, Li Zheng, Na Li, Weiping Wang, Shouyan Zhong, and Kaishun Hu. Minimizing mean weighted tardiness in unrelated parallel machine schedul-

ing with reinforcement learning. *Computers & operations research*, 39(7):1315–1324, 2012.

[13] Zhicong Zhang, Li Zheng, and Michael X Weng. Dynamic parallel machine scheduling with mean weighted tardiness objective by q-learning. *The International Journal of Advanced Manufacturing Technology*, 34(9-10):968–980, 2007.

[14] Zhicong Zhang, Weiping Wang, Shouyan Zhong, and Kaishun Hu. Flow shop scheduling with reinforcement learning. *Asia-Pacific Journal of Operational Research*, 30(05):1350014, 2013.

[15] M Emin Aydin and Ercan Öztemel. Dynamic job-shop scheduling using reinforcement learning agents. *Robotics and Autonomous Systems*, 33(2-3):169–178, 2000.

[16] Thomas Gabel and Martin Riedmiller. Distributed policy search reinforcement learning for job-shop scheduling tasks. *International Journal of production research*, 50(1):41–61, 2012.

[17] Yi-Chi Wang and John M Usher. A reinforcement learning approach for developing routing policies in multi-agent production scheduling. *The International Journal of Advanced Manufacturing Technology*, 33(3-4):323–333, 2007.

[18] Biao Yuan, Lei Wang, and Zhibin Jiang. Dynamic parallel machine scheduling using the learning agent. In *2013 IEEE International Conference on Industrial Engineering and Engineering Management*, pages 1565–1569. IEEE, 2013.

[19] Shuhui Qu, Jie Wang, Shivani Govil, and James O. Leckie. Optimized Adap-

tive Scheduling of a Manufacturing Process System with Multi-skill Workforce and Multiple Machine Types: An Ontology-based, Multi-agent Reinforcement Learning Approach. *Procedia CIRP*, 57:55–60, January 2016.

[20] Kfir Arviv, Helman Stern, and Yael Edan. Collaborative reinforcement learning for a two-robot job transfer flow-shop scheduling problem. *International Journal of Production Research*, 54(4):1196–1209, February 2016.

[21] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015.

[22] B. Waschneck, A. Reichstaller, L. Belzner, T. Altenmüller, T. Bauernhansl, A. Knapp, and A. Kyek. Deep reinforcement learning for semiconductor production scheduling. In *2018 29th Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC)*, pages 301–306, April 2018.

[23] Huan Xu and Shie Mannor. Robustness and Generalization. May 2010. arXiv: 1005.2243.

[24] Chiyuan Zhang, Oriol Vinyals, Remi Munos, and Samy Bengio. A Study on Overfitting in Deep Reinforcement Learning. April 2018. arXiv: 1804.06893.

[25] Jesse Farebrother, Marlos C. Machado, and Michael Bowling. Generalization and Regularization in DQN. September 2018. arXiv: 1810.00123.

[26] J. Lim, M. Chae, Y. Yang, I. Park, J. Lee, and J. Park. Fast Scheduling of Semiconductor Manufacturing Facilities Using Case-Based Reasoning. *IEEE Transactions on Semiconductor Manufacturing*, 29(1):22–32, February 2016.

[27] Jaeseok Huh, Inbeom Park, Seongmin Lim, Bohyung Paeng, Jonghun Park, and Kwanho Kim. Learning to Dispatch Operations with Intentional Delay for Re-Entrant Multiple-Chip Product Assembly Lines. *Sustainability*, 10(11):4123, November 2018.

[28] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, November 2018.

[29] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, May 1992.

[30] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[31] Jan Kukačka, Vladimir Golkov, and Daniel Cremers. Regularization for Deep Learning: A Taxonomy. October 2017. arXiv: 1710.10686.

[32] Liji Shen, Stéphane Dauzère-Pérès, and Janis S. Neufeld. Solving the flexible job shop scheduling problem with sequence-dependent setup times. *European Journal of Operational Research*, 265(2):503–516, March 2018.

[33] Guohui Zhang, Liang Gao, and Yang Shi. An effective genetic algorithm for the flexible job-shop scheduling problem. *Expert Systems with Applications*, 38(4):3563–3573, April 2011.

[34] Yong Ming Wang, Hong Li Yin, and Kai Da Qin. A novel genetic algorithm for flexible job shop scheduling problems with machine disruptions. *The International Journal of Advanced Manufacturing Technology*, 68(5):1317–1326, September 2013.

[35] Yailen Martínez, Ann Nowé, Juliett Suárez, and Rafael Bello. A Reinforcement Learning Approach for the Flexible Job Shop Scheduling Problem. In Carlos A. Coello Coello, editor, *Learning and Intelligent Optimization*, Lecture Notes in Computer Science, pages 253–262. Springer Berlin Heidelberg, 2011.

[36] George Konidaris and Andrew G. Barto. Autonomous shaping: Knowledge transfer in reinforcement learning. volume 148, pages 489–496, January 2006.

[37] Felipe Leno Da Silva and Anna Helena Reali Costa. Transfer Learning for Multiagent Reinforcement Learning Systems. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 3982–3983. AAAI Press, 2016.

[38] Matthew E. Taylor and Peter Stone. Cross-domain Transfer for Reinforcement Learning. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 879–886, New York, NY, USA, 2007. ACM.

[39] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A Survey on Deep Transfer Learning. August 2018. arXiv: 1808.01974.

[40] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. March 2017. arXiv: 1703.03400.

[41] Dino S. Ratcliffe, Luca Citi, Sam Devlin, and Udo Kruschwitz. Domain Adaptation for Deep Reinforcement Learning in Visually Distinct Games. February 2018.

[42] Thomas Carr, Maria Chli, and George Vogiatzis. Domain Adaptation for Reinforcement Learning on the Atari. December 2018. arXiv: 1812.07452.

[43] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M. Hospedales. Learning to Generalize: Meta-Learning for Domain Generalization. October 2017. arXiv: 1710.03463.

[44] Todd Hester and Peter Stone. Generalized model learning for reinforcement learning in factored domains. In *AAMAS*, 2009.

[45] Inbeom Park, Jaeseok Huh, Joongkyun Kim, and Jonghun Park. Setup change scheduling for semiconductor manufacturing facilities using multi-agent reinforcement learning. *unpublished*.

[46] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013.

[47] Peter J. Huber. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, March 1964.

[48] Yogesh Balaji, Swami Sankaranarayanan, and Rama Chellappa. MetaReg: Towards Domain Generalization using Meta-Regularization. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Ad-*

vances in Neural Information Processing Systems 31, pages 998–1008. Curran Associates, Inc., 2018.

[49] R. Haupt. A survey of priority rule-based scheduling. *Operations-Research-Spektrum*, 11(1):3–16, March 1989.

[50] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. December 2014. arXiv: 1412.6980.

[51] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

[52] Prof. Dr. Imed Kacem, Slim Hammadi, and Pierre Borne. Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. *IEEE Transactions on Systems Man and Cybernetics Part C (Applications and Reviews)*, 32:1–13, February 2002.

[53] Parviz Fattahi, Mohammad Saidi Mehrabad, and Fariborz Jolai. Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. *Journal of Intelligent Manufacturing*, 18(3):331–342, June 2007.

[54] Jie Gao, Mitsuo Gen, Linyan Sun, and Xiaohui Zhao. A hybrid of genetic algorithm and bottleneck shifting for multiobjective flexible job shop scheduling problems. *Computers & Industrial Engineering*, 53(1):149–162, August 2007.

Abstract

Enhancing Robustness of Deep Reinforcement Learning based Semiconductor Packaging Lines Scheduling with Regularized Training

Joongkyun Kim
Department of Industrial Engineering
The Graduate School
Seoul National University

As the demand for high-performance electronic devices has increased, the semiconductor manufacturing process is being developed centering on the production of multi-chip products. In multi-chip products, re-entrance occurs by repeating the process several times in the packaging line, and the setup change of equipment is frequently incurred. These are major factors that make the scheduling of the semiconductor packaging line difficult. The production environment frequently changes due to internal and external variabilities. In addition, since the calculation time required for scheduling is very important at the manufacturing site, prompt schedule generation is required. As the research of the semiconductor packaging line scheduling becomes active, the reinforcement learning based scheduling research aiming at

the global optimization is increasing. In view of the utilization of scheduling research based on reinforcement learning, there is a need for a method capable of reacting to various production environment changes and obtaining a good schedule in a short time.

This study aims at obtaining the robustness of the scheduling model based on deep reinforcement learning. We propose a regularization training method for semiconductor packaging lines scheduling based on deep reinforcement learning without performance degradation and re-training when a new production environment is given as a test data. In order to apply reinforcement learning to flexible job-shop scheduling problem, we designed state, action and reward considering overall process and trained deep Q network which is a representative algorithm of deep reinforcement learning. The regularization training method proposed in this study is divided into four stages and designed to train the generalities of the problems reflected in various production environment and the specificity of each problem. Experiments were conducted using scheduling problems of different complexity, and it was verified that the performance was superior to other scheduling models based on rule-based and deep reinforcement learning.

This study is the first research that focuses on the robustness of the model in the reinforcement learning based scheduling. Moreover, the result of this study enhances the practicality of research in real factory application.

Keywords: Semiconductor packaging lines, Flexible job-shop scheduling, Deep reinforcement learning, Regularized training

Student Number: 2017-29546